COMMODORE C 16

Guida alla programmazione in BASIC 3.5

di BRIAN LLOYD

LIBRO + CASSETTA

COMMODORE C 16

Guida alla programmazione in BASIC 3.5

di BRIAN LLOYD

Traduzione di F. FRANCIA
e T. POLICASTRO

JACOPO CASTELFRANCHI EDITORE
Via dei Lavoratori, 124
CINISELLO BALSAMO (MI)

INDICE

Introduzione **Pagina**

Indice dettagliato dei capitoli

Introduzione

Questo libro è rivolto a chi è del tutto nuovo nel mondo dei computer, e con questo spirito è stato scritto l'intero testo. Ciò non significa però che anche il programmatore già esperto non possa per suo tramite acquisire le necessarie conoscenze sulla versione 3.5 del BASIC che è stata adottata dai computer Commodore C16 e Plus/4, le cui istruzioni illustra in dettaglio. Imparerete così ad usare il vostro computer, e come fargli fare quello che desiderate. Ogni concetto è esaurientemente spiegato, dai fatti più semplici come l'effettuazione dei collegamenti iniziali, sino alle tecniche di programmazione più sofisticate, come la gestione dei file o gli impieghi di PEEK e POKE.

Anche se ogni cosa è spiegata in dettaglio, si è cercato di non indulgere in lunghe e non necessarie illustrazioni dei concetti più semplici. Il libro è stato invece concepito in modo da consentirvi di scrivere i primi programmi in breve tempo. Le diverse istruzioni vengono presentate a mano a mano che servono, in modo da non confondervi con comandi troppo complessi prima ancora che siate in grado di padroneggiare la tastiera.

Vi consigliamo caldamente di leggere con cura ogni parte del libro, senza saltare qualche pagina perchè appare noiosa. Sappiamo che si tratta di una cosa difficile da chiedere, perchè ognuno tende a voler imparare presto ed a provare subito qualcosa di nuovo. Purtroppo così facendo si imboccano spesso dei vicoli ciechi, per cui sarà meglio che procediate lentamente e metodicamente nella lettura del libro, se veramente desiderate diventare un buon programmatore. Se qualcosa non la capite subito, tornate indietro e rileggete la parte interessata.

La cosa più importante è provare direttamente, da sempre il miglior modo per imparare. Sperimentate ogni cosa che avete appena imparato, per vedere come funziona (ed anche a volte perchè non funziona!). Non temete di arrecare qualche danno al computer così facendo, perchè, a meno di colpirlo con un martello, è davvero difficile che possiate danneggiarlo.

Infine, prima che cominciate, potrà interessarvi sapere che il linguaggio adottato dal vostro computer, e che questo libro intende insegnavi, si chiama BASIC. Questa sigla sta per Beginners All-purpose Symbolic Instruction Code (Codice simbolico di istruzioni multi-uso per principianti): anche se potrà sembrarvi un po' complicato, dovrebbe rassicurar-

vi la presenza nella definizione di BASIC della parola "principianti". D'altra parte il termine non vi tragga in errore: vi assicuriamo che si tratta di un linguaggio estremamente potente.

Ringraziamenti

Desidero qui ringraziare l'intero "staff" della Commodore Business Machines (UK) Ltd, ed in particolare Gail Wellington, Phil Gosling, e tutti i tecnici, senza la cui paziente collaborazione non avrei potuto scrivere questo libro.

Per cominciare

I collegamenti

La prima e più importante cosa da imparare per poter usare il vostro Commodore C16 o Plus/4 consiste nell'effettuare i giusti collegamenti. Il vostro computer va collegato al suo alimentatore e ad un ricevitore TV, e successivamente ad un registratore a cassette, ed eventualmente ad un'unità disco e ad una stampante, cose che tratteremo più avanti.

La prima cosa da fare è collegare l'alimentatore. La relativa spina va inserita in una presa elettrica idonea. Lo spinotto collegato al filo che esce dall'alimentatore va a sua volta inserito nella presa femmina marcata "POWER" (che si trova sulla sinistra del Plus/4, visto da dietro, ovvero sulla destra del C16 sempre sul retro), dopo essersi accertati che l'interruttore di accensione del vostro computer sia in posizione di "off" (spento).

Col computer vi viene pure fornito un cavetto coassiale da collegare al TV. Ad un suo estremo c'è uno spinotto tipo "fono", che dovete inserire nella presa marcata "RF" (posta sul lato sinistro del Plus/4, e sul retro del C16): l'altro spinotto va inserito invece nella presa di antenna del televisore.

Per il momento non ci sono altri collegamenti da fare, dato che il registratore, o l'unità disco, o la stampante verranno collegati al momento che imparerete come servirvene.

Ed ecco il grande momento! Accendete il televisore, e poi anche il computer (si illuminerà una spia rossa) e..., a meno che non siate veramente fortunati, sullo schermo avrete la stessa immagine che di solito compare quando l'antenna non è collegata, mentre i vostri orecchi saranno feriti da un fruscio più o meno assordante. In effetti, a questo punto dovete procedere a sintonizzare correttamente il televisore, in prossimità del canale 36. Se agirete con cura, vedrete comparire sullo schermo un messaggio, e cioè

COMMODORE BASIC V3.5 60671 BYTES FREE

READY

nel caso del Plus/4, mentre con il C16 il messaggio sarà

COMMODORE BASIC V3.5 12277 BYTES FREE

READY

Si tratta del messaggio di inizializzazione, che vi informa che il vostro computer è in grado di comprendere la versione 3.5 del Commodore BASIC, ed inoltre vi segnala l'ammontare (in byte) di memoria libera disponibile (un byte può contenere un carattere, cosicchè p. es. la parola SALVE necessita di 5 byte perchè composta di 5 caratteri). Il termine READY ("pronto") vi dice che il computer attende che gli diciate cosa fare. Sotto READY vedrete un quadratino lampeggiante, che viene detto "cursore". Il cursore serve ad indicare la posizione sullo schermo in cui sarà visualizzato ("stampato") il successivo carattere che verrà battuto sulla tastiera.

Se possedete un Plus/4 vi verrà anche segnalato che è inserita una ROM 3-PLUS-1 e quale tasto premere per poterla usare. Consultate il vostro manuale 3-PLUS-1 per i particolari sull'impiego di questo ottimo accessorio.

La tastiera

Ed ora vediamo la tastiera. A prima vista può apparire complicata, ma vedrete che non ci vorrà molto perchè siate in grado di usarla correttamente. Quelli fra di voi già adusati ad una tastiera per macchina da scrivere noteranno che i tasti delle lettere seguono la familiare disposizione QWERTY (ossia, la prima fila di tasti letterali comincia con le lettere Q-W-E-R-T-Y). Tuttavia i tasti del vostro computer recano un numero di simboli molto maggiore che nelle comuni macchine per scrivere, e ci sono anche dei tasti extra.

Normalmente, usando una macchina per scrivere, le lettere corrispondenti ai tasti premuti compaiono su un foglio di carta, in minuscolo. Se battete su una tastiera di computer invece le lettere compaiono sullo schermo, in maiuscolo. Se battete

SALVE A TUTTI

vedrete comparire le lettere di questo testo in alto sullo schermo, in caratteri maiuscoli, ed il cursore spostarsi in successione da sinistra verso destra.

Lettere minuscole

Il computer è tuttavia in grado di produrrre anche le minuscole. Tenete premuto uno dei due tasti marcati SHIFT e contemporaneamente preme-

te il tasto "Commodore" (quello in basso a sinistra con la scritta Commodore) vedrete che le lettere visualizzate d'ora in poi sullo schermo diventano minuscole. Se battete

salve a tutti

questa volta il messaggio comparirà sullo schermo in minuscole.
Quando scrivete su di una macchina per scrivere e volete una lettera maiuscola, sapete che dovete contemporaneamente tenere abbassato il tasto SHIFT mentre premete il tasto. Lo stesso vale per la tastiera del computer quando vi trovate in modalità "minuscole". Per esempio, se volete stampare

Salve Mario

dovete accertarvi innanzitutto di trovarvi in modalità minuscole, poi premere il tasto SHIFT mentre premete il tasto S, quindi rilasciare lo SHIFT e battere alve, seguito da uno spazio (premendo la barra spaziatrice); quindi ripremere SHIFT assieme al tasto M, infine le lettere di ario.

Quando dovrete introdurre delle istruzioni, vedrete che potrete farlo impostando sia caratteri minuscoli che maiuscoli. Tuttavia, se vi trovate in modalità minuscole e premete un tasto assieme a SHIFT nell'impostare un comando, questo comando verrà ignorato.

Simboli con lo SHIFT

Il tasto SHIFT si può pure usare per ottenere i simboli posti sopra i numeri. Per esempio, se si vuole ottenere il simbolo $ ("dollaro") si deve tenere premuto il tasto SHIFT e premere il tasto 4. Lo stesso vale per i tasti coi simboli della virgola, del punto, "/", ":",";", ognuno dei quali reca due simboli sullo stesso tasto. Per ottenere uno dei simboli posti nella parte superiore del tasto occorre premere il tasto SHIFT assieme al tasto con il simbolo voluto.

"Pulizia" dello schermo

A questo punto lo schermo si sarà un bel pò riempito. Fortunatamente, è assai facile "ripulire" lo schermo di tutto il suo contenuto: se si preme il tasto marcato CLEAR/HOME, il cursore salterà nell'angolino superiore sinistro dello schermo. Se lo stesso tasto lo premete assieme al tasto SHIFT, lo schermo verrà cancellato, ed ancora il cursore si piazzerà nell'angolo superiore sinistro, pronto per ricominciare.

I tasti di controllo del cursore

Avrete probabilmente notato i quattro tasti che recano sopra delle frecce. Essi vengono chiamati tasti di controllo per il movimento del cursore, perchè la pressione di ciascuno di essi provoca lo spostamento del cursore di un passo nella direzione indicata dalla freccia. Con questi tasti potete scegliere il punto dove volete vedere comparire il prossimo carattere.

Il tasto INST/DEL

Nell'angolo superiore destro della tastiera c'è un tasto con la dicitura INST/DEL. Con questo tasto potete cancellare (DELete) singoli caratteri dallo schermo, nonchè INSerire spazi fra caratteri. Per esempio, dopo esservi accertati di essere in modalità maiuscole (ricordate? potete commutare fra minuscole e maiuscole premendo assieme il tasto Commodore e lo SHIFT), battete

CARLO ANDAAVA STRUDA

Per prima cosa dobbiamo correggere STRUDA. Portatevi col cursore sulla lettera U (mediante i tasti di controllo del cursore), poi battete ADA: queste lettere si sovrapporranno a quelle precedenti, ed avremo

CARLO ANDAAVA STRADA

Ora dobbiamo eliminare una A di troppo dalla seconda parola. Per fare questo, posizionate il cursore sulla lettera V che segue la A di troppo, e premete il tasto INST/DEL. La seconda A sparirà, lasciando

CARLO ANDAVA STRADA

che però non va ancora bene: dobbiamo inserire PER fra le parole ANDAVA e STRADA per avere un senso compiuto. Posizionate il cursore sulla lettera iniziale del secondo termine (S). Se ora tenete premuto il tasto SHIFT mentre premete il tasto INST/DEL vedrete che STRADA si sposta di 6 spazi verso destra. Ora potete rilasciare il tasto SHIFT e battere le lettere PER (seguite da uno spazio): così avremo infine la frase

CARLO ANDAVA PER STRADA

Il tasto SHIFT LOCK

Sopra il tasto SHIFT sulla sinistra trovate un tasto con la dicitura SHIFT LOCK: esso serve a bloccare lo SHIFT, proprio come se lo manteneste sempre premuto con un dito, evitandovi questa fatica quando vi servono più caratteri successivi da battere assieme a SHIFT. Se premete SHIFT LOCK una volta, sentirete un click e noterete che esso si fissa in posizione semiabbassata,ad indicare che ora è attivato. Per eliminare questo effetto, basta premere SHIFT LOCK una seconda volta.

Il tasto CONTROL

Un altro tasto con una speciale funzione è il tasto marcato CONTROL. Il Plus/4 è dotato di due di questi tasti CONTROL, posti ai due estremi della seconda fila di tasti. Il C16 dispone di un solo tasto CONTROL, ma non è poi un grande svantaggio. Se osservate i tasti numerici con le cifre da 1 a 8, vedrete che ciascuno reca sopra due colori. Tenendo premuto il tasto CONTROL mentre si preme uno di questi 8 tasti potete modificare il colore del cursore (nonchè il colore di qualsiasi testo che verrà successivamente battuto) in quello indicato superiormente sul tasto.

Il tasto Commodore

Il tasto Commodore può essere usato per selezionare i colori scritti in basso sui tasti numerici. Per es., se premete il tasto Commodore assieme al tasto "4" il colore del cursore passerà al rosa.

Caratteri invertiti

Il tasto CONTROL può anche venire utilizzato per ottenere caratteri "invertiti". Per intenderci, se un testo normale appare sullo schermo in bianco su sfondo nero, a caratteri invertiti apparirà in nero su fondo bianco. La cosa migliore è di provare, battendo qualche carattere, e poi premere assieme il tasto CONTROL ed il tasto 9 (che reca la dicitura RVS ON). Rilasciate entrambi i tasti, e battete qualche altro carattere: potrete così vedere direttamente come appaiono i caratteri invertiti. Per tornare in condizioni normali, basta premere assieme il tasto CONTROL ed il tasto 0 (con la dicitura RVS OFF).

Caratteri lampeggianti

Un altro uso del tasto CONTROL è per ottenere caratteri lampeggianti. Se tenete premuto il tasto CONTROL e premete il tasto che reca la

virgola (nonchè la dicitura "Flash On"), ogni carattere che batterete successivamente apparirà lampeggiante. Premendo CONTROL assieme al tasto con il punto (dicitura "Flash Off") i caratteri che batterete da quel momento in avanti saranno normali, ma i caratteri lampeggianti rimarranno tali.

Simboli grafici

Sulla parte anteriore di vari tasti avrete probabilmente notato alcuni strani simboli formati da linee e curve. Si tratta dei simboli grafici, che vengono ottenuti tramite il tasto SHIFT e quello Commodore. Una volta accertato che il computer è in modalità maiuscole, premete il tasto SHIFT (oppure SHIFT LOCK) e, tenendolo abbassato, provate a premere qualcuno dei tasti che reca un simbolo grafico. Potrete notare che con la pressione di SHIFT ottenete i simboli posti sulla destra. Se rilasciate SHIFT (o sbloccate SHIFT LOCK) e tenete invece premuto il tasto Commodore mentre premete i tasti dei simboli grafici, questa volta otterrete quelli di sinistra.

Se ora commutate sulle minuscole, vedrete che sullo schermo alcuni dei simboli grafici già tracciati passerano a lettere maiuscole. Ciò dipende dal fatto che in modalità minuscole il tasto SHIFT serve per ottenere le maiuscole, e non i simboli grafici. In altre parole, se siete in modalità minuscole, perdete tutti i simboli grafici di destra e, se premete uno dei tasti dei simboli grafici assieme a SHIFT ottenete invece lettere maiuscole.

Il tasto RETURN

Può essere capitato, nel corso dei vostri esperimenti precedenti, di avere inavvertitamente premuto il tasto RETURN, nel qual caso sarà comparso il messaggio "?SYNTAX ERROR". Se ciò accade mentre state sperimentando con la tastiera, niente paura. Il tasto RETURN indica al computer di passare ad eseguire qualsiasi istruzione abbiate impostato, per cui se avete battuto "SALVE" e premuto il tasto RETURN, comparirà il messaggio citato, perchè il computer non è in grado di interpretare come istruzione la parola SALVE.

Restano ormai solo sei tasti che non conoscete ancora: i quattro tasti funzione (da F1 a F8), il tasto ESC ed il tasto RUN/STOP. Ve li illustreremo più avanti.

Il pulsante di RESET

Sul lato destro del vostro computer trovate un piccolo pulsante marcato "RESET". Premendo questo pulsante si ottiene la cosiddetta "reinizializzazione a freddo" (il che significa che i contenuti della memoria vengono persi), con la comparsa del consueto messaggio iniziale. Se però tenete premuto il tasto RUN/STOP e premete questo pulsante, lo schermo vien cancellato e compare un messagio "MONITOR" seguito da una serie di lettere e numeri. Se ora battete X e premete RETURN, ogni cosa tornerà normale, e il programma eventualmente presente in memoria sarà ancora disponibile (Si è realizzata una "reinizializzazione a caldo").

Sommario

Eccovi una breve sintesi del funzionamento dei diversi tasti speciali:

SHIFT: serve ad ottenere determinati simboli (come $, &, []; per ottenere i simboli grafici di destra sui tasti e, se usato assieme al tasto Commodore, alla commutazione fra minuscole e maiuscole.

SHIFT LOCK: mantiene attivo il tasto SHIFT.

COMMODORE: per ottenere i colori "inferiori" segnati sui tasti numerici (da 1 ad 8); per ottenere i simboli grafici di sinistra e, assieme al tasto SHIFT, per commutare come detto fra minuscole e maiuscole.

CLEAR/HOME: serve per riportare il cursore nella posizione iniziale sullo schermo (angolo superiore sinistro). Se usato assieme a SHIFT serve a cancellare tutto quello che c'è sullo schermo.

INST/DEL: per cancellare dei caratteri. Assieme a SHIFT serve per inserire degli spazi fra caratteri.

CONTROL: per ottenere i colori "superiori" indicati sui tasti numerici (da 1 a 8) e, assieme ai tasti 9 e 0, per attivare o disattivare i caratteri invertiti. Usato anche assieme ai tasti con la virgola ed il punto per attivare e disattivare i caratteri lampeggianti.

TASTI CON LE FRECCE: servono per lo spostamento del cursore sullo schermo.

L'istruzione PRINT

A questo punto dovreste aver preso sufficiente confidenza con la tastiera, e sapere pressappoco dove sta ciascun carattere: siete ormai pronti a far lavorare il vostro computer per voi.

Uno dei compiti principali che un computer è capace di svolgere è l'esecuzione di calcoli. Per poter fare dei calcoli dovete saper usare il comando (istruzione) PRINT.

Supponiamo, per esempio, che vogliate calcolare quanto fa 9+7. Per fare questo, dovete battere questo comando:

```
PRINT 9+7
```

e poi premere il tasto di RETURN. La pressione di questo tasto segnala al computer che deve passare ad eseguire il comando che avete battuto, in questo caso l'addizione di 9 con 7, e visualizzare il risultato sullo schermo.

Se invece della riposta vedete comparire un messaggio di errore, significa che avete fatto qualche errore di battitura. In tal caso, basterà che spostiate il cursore nella posizione dove sta l'errore, lo correggiate (allo stesso modo con cui avete provveduto a correggere una frase nel capitolo precedente), e premiate nuovamente il tasto RETURN.

Quando avete impostato il comando citato, voi avete ordinato al computer di PRINT ("stampare", qui visualizzare sullo schermo) il risultato di 9+7. Naturalmente il computer è capace di eseguire calcoli più complessi di questa semplice addizione. Provate con gli esempi che seguono (ricordandovi di premere sempre RETURN al termine dell'impostazione di ciascun comando):

```
PRINT 32-17
PRINT 9*54
PRINT 99/11
PRINT 1564+1928
```

Da questi calcoli avrete stabilito che il segno * significa "per" (segno di moltiplicazione) ed il segno / significa "diviso per". Questi simboli sono

comuni a quasi tutti i computer. Provate ad eseguire alcuni calcoli di testa vostra e verificare i risultati.
Se avete provato ad eseguire calcoli multipli come 92+8/4, nel vederne il risultato potreste aver pensato che il computer si sia sbagliato. Provate ora con

PRINT 6+4/2

Sarete forse sorpresi di trovare che la risposta è 8 e non 5. Ciò dipende dal fatto che i computer svolgono alcuni calcoli con priorità rispetto a certi altri. Nel caso del vostro computer (e di molti altri) i prodotti e le divsioni vengono svolti con priorità rispetto alle somme ed alle sottrazioni. Ciò significa che nel calcolare quanto sopra, il computer esegue la somma di 6+ (il risultato di 4/2), il che dà ovviamente 8. Se si voleva la risposta 5 il calcolo da eseguire andava scritto

PRINT (6+4)/2

Ponendo delle parentesi attorno a "6+4" si dice al computer che deve prima eseguire questa parte del calcolo, per cui questo dà come risultato (la somma di 6+4) diviso per 2, ossia 5.
Avrete probabilmente notato che sul tasto dello 0 c'è una freccia. Questo simbolo significa "elevazione a potenza", per cui battendo

PRINT 3↑4

si ottiene sullo schermo la risposta 81, perchè 3 alla 4.a potenza vale (3 per 3 per 3 per 3) = 81.

L'elevazione a potenza ha la massima priorità, ossia viene eseguita prima di qualsiasi altra operazione. Le priorità vanno quindi nell'ordine (decrescente)

elevazione a potenza, prodotto, divisione, addizione, e sottrazione

Se per fare un esempio il computer deve calcolare il risultato di 5+7-2, esso eseguirà i calcoli nell'ordine in cui sono scritti, perchè addizione e sottrazione hanno la stessa priorità, così come il prodotto e la divisione.

Si possono anche eseguire calcoli con numeri negativi. Basta inserire un segno "meno" davanti ad un numero, per denotare che si tratta di un numero negativo. Per esempio

PRINT -2+7

darà per risultato 5.

Sommario

Per eseguire calcoli si usa l'istruzione PRINT, nel formato PRINT [espressione]. I calcoli vengono eseguiti nell'ordine di priorità

Elvazione a potenza
Moltiplicazione o divisione
Somma o sottrazione

Se il calcolo comprende, p. es., una sottrazione seguita da un'addizione, la sottrazione viene eseguita per prima, perchè queste due operazioni hanno lo stesso grado di priorità.

Si possono porre delle parentesi attorno ad alcuni elementi di un calcolo, per fare sì che il computer esegua tale parte prima delle altre, indipendentemente da altre priorità.

Stampa di testi

Una delle altre funzioni svolte dall'istruzione (comando) PRINT è la visualizzazione di testi sullo schermo. Se battete

PRINT"SALVE, COME STAI?"

(seguito ovviamente da RETURN) vedrete comparire sullo schermo il messaggio SALVE, COME STAI?.

Con l'istruzione PRINT si può visualizzare qualsiasi testo che sia compreso fra due coppie di virgolette ("). Le virgolette segnalano al computer che quanto sta fra di esse non è un calcolo, ma esattamente il testo che volete visualizzare.

Modifica del colore dei testi

Può risultare utile visualizzare parti diverse di testo in colori differenti. Per ottenere questo effetto occorre inserire i controlli per il colore nelle istruzioni di PRINT usate per visualizzare i testi. La cosa è abbastanza facile. Impostate normalmente il comando di PRINT e poi, là dove desiderate modificare il colore, tenete abbassato il tasto Commodore o quello CONTROL (dipende dal colore desiderato) e premete il tasto numerico che reca il colore prescelto per il testo successivo. Ad esempio, se volete visualizzare il testo SALVE FRANCO con SALVE scritto in rosso e FRANCO in blu, dovete procedere esattamente come segue (dove trovate qualcosa fra parentesi quadre significa che dovete seguire le istruzioni, ma non battere quanto c'è fra le parentesi stesse!):

PRINT"[tenete premuto il tasto CONTROL e premete il tasto 3, poi rilasciate ambedue i tasti]SALVE[tenete premuto il tasto CONTROL e premete il tasto 7, poi rilasciateli entrambi]FRANCO".

Quando avrete premuto RETURN vedrete comparire il messaggio colorato sullo schermo. Come potete notare, quando modificate un colore di stampa in un'istruzione PRINT, compare un carattere invertito: ciò serve a rammentarvi il punto dove avete modificato il colore, e qual'è questo colore.

Abbreviazione per PRINT

Può essere utile sapere che invece di scrivere PRINT ogni volta per esteso potete usare semplicemente il simbolo ?: il computer interpreterà il ? come PRINT ed eseguirà le stesse azioni che avreste ottenuto impostando il comando per intero. Per esempio, battendo ?"SALVE" si ha il medesimo risultato di PRINT"SALVE": il fatto è utile perchè risparmia un sacco di digitazione, e quindi farete bene a ricordarlo ed usarlo.

Così avete compiuto i primi passi nell'apprendimento della programmazione. Avete imparato come fare eseguire certi semplici calcoli al vostro computer (con l'uso dell'istruzione PRINT), ed anche come visualizzare varie parti di testo in diversi colori. Sapete ormai usare a dovere la tastiera, ed avete imparato inoltre che il computer passa ad eseguire quanto richiesto solo dopo aver premuto il tasto RETURN.

Sommario

Tutto quello che segue un comando PRINT ed è posto fra virgolette verrà visualizzato sullo schermo esattamente come battuto.

Per modificare il colore dei testi inclusi in un'istruzione PRINT si procede esattamente come quando si vogliono modificare normalmente i colori. Verrà visualizzato un carattere invertito ad indicare da dove si è modificato il colore di stampa, e di quale colore si tratta.

Il comando PRINT può essere abbreviato in ?.

Sarebbe bene a questo punto che vi prendiate una mezz'oretta di tempo per sperimentare quello che avete imparato sin qui, e vi accertiate di aver ben compreso ogni cosa. In tal modo vi risparmierete di dover tornare spesso indietro nel testo: e potrete ben presto scrivere il vostro primo programma in BASIC.

Il vostro primo programma

I numeri di linea

Probabilmente avrete già sentito parlare di "programmi per computer", e sarete curiosi di sapere di cosa esattamente si tratta.

Un programma per computer è una successione di istruzioni che vengono eseguite in un ordine determinato da parte del computer. Sino ad ora, avete semplicemente ordinato al computer di svolgere immediatamente i vari comandi che gli avete dato : si tratta del cosiddetto modo "immediato" (COMMAND MODE): ogni volta che il computer esegue un comando in modo immediato esso lo "dimentica" completamente. Quando scrivete un programma non è questo che volete, in quanto le varie istruzioni non vanno eseguite appena terminate di impostare. Per ovviare a questo problema, nello scrivere dei programmi si usano i numeri di linea. Il che ci consente inoltre di rieseguire più e più volte il medesimo programma, ogni volta che lo desideriamo.

I numeri di linea vengono usati per far sì che il computer ricordi le singole istruzioni che gli avete fornito, e per indicargli l'ordine in cui queste devono essere eseguite. Molti preferiscono che i numeri di linea si succedano ad incrementi regolari di 10 (p. es. 10, 20, 30,...), perchè in questo modo si lasciano ampi spazi nel programma per l'inserimento di eventuali ulteriori linee, come vedremo fra un momento.

Ed ecco il vostro primo programma. Come potete osservare, ogni singola istruzione è posta su di una linea diversa, con un numero di linea. Provate a battere il programma esattamente come sta (ricordandovi di premere RETURN al termine di ciascuna linea!):

```
10SCNCLR
20PRINT"SALVE!!!"
30PRINT"QUESTO È IL VOSTRO PRIMO PROGRAMMA?"
40PRINT"MAGARI NON È PROPRIO ECCEZIONALE..."
50PRINT"PERO' I PROGRAMMI MIGLIORERANNO
        COL TEMPO!"
```

Per la verità, non è successo un gran che...Ciò dipende dal fatto che tutto

si è sinora ridotto all'introduzione di un programma che il computer è in grado di ricordare, almeno sino a quando non gli direte di scordarlo. Per far "lavorare" il programma dobbiamo battere il comando RUN (seguito come al solito da RETURN). Non appena fatto questo, il computer passerà a svolgere il programma (RUN significa "parti", ovvero "esegui").

In certi casi potrete veder comparire un "messaggio di errore": il computer vi segnalerà allora in quale linea l'errore si è verificato. In questi casi dovrete reimpostare quella linea.

SCNCLR

Avrete notato che nella linea 10 compare un nuovo comando: SCNCLR. Esso indica al computer di "ripulire" lo schermo, allo stesso modo di quando premete in modo diretto SHIFT ed il tasto CLEAR/HOME. Il resto del programma si compone di istruzioni PRINT, e dovreste quindi essere in grado di capire come funzionano. In alternativa al comando SCNCLR si può eseguire il PRINT di un simbolo "cuore", che si ottiene premendo il tasto SHIFT assieme al tasto CLEAR/HOME.

Bene, ora abbiamo memorizzato il programma, e l'abbiamo anche fatto eseguire: e poi? Una delle altre cose che si possono fare è di "listare" sullo schermo il programma impostato, battendo LIST (e come sempre facendo seguire RETURN). In questo modo vedrete comparire sullo schermo il "listato" del programma, linea dopo linea. Il tutto avviene molto velocemente.

Una delle cose che avrete forse notato è che quando si dà il RUN (si "lancia") il programma, ogni istruzione PRINT stampa il proprio testo su di una linea separata. Tuttavia, è possibile far continuare il testo di un PRINT là dove era terminato il precedente. Per capire meglio, aggiungete al programma queste due altre linee:

```
60PRINT"3232+5674=";
70PRINT3232+5674
```

Se ora date il RUN al programma, vedrete comparire gli stessi messaggi di prima, ma alla fine comparirà una linea che reca '3232+5674=8906'.

La linea 60 del programma dice al computer di visualizzare i caratteri '3232+5674=', e la linea 70 di elaborare il risultato del calcolo di 3232+5674 e di visualizzarlo sullo schermo. Il punto importante consiste nel fatto che il secondo PRINT fa visualizzare il risultato (8906) sulla

stessa linea in cui il primo PRINT aveva mostrato il calcolo: perchè? La risposta sta nel punto-e-virgola (;) che vedete scritto alla fine della linea 60: è questo simbolo che segnala al computer di non iniziare la nuova linea di stampa su di una riga separata, ma di continuare dal punto dove finiva la stampa precedente.

Forse vi ricordate che all'inizio del capitolo ho menzionato che molti preferiscono numerare le linee dei loro programmi con incrementi di 10, permettendo così l'eventuale inserimento di nuove linee in un secondo tempo. Per comprendere cosa significa ciò, provate a battere la linea che segue terminata dal solito RETURN, e poi date il comando LIST:

55PRINT"ESEMPIO DI CALCOLO: "

Quando il listato del programma comparirà sullo schermo, vedrete che la nuova linea (la linea 55) si sarà inserita fra la linea 50 e la 60. Provate a dare RUN al programma, e vedete l'effetto portato dalla nuova linea.

Se nell'impostare un programma avete commesso un errore, per correggerlo dovete ribattere da capo la linea sbagliata. Il motivo è che quando si assegna ad una nuova linea lo stesso numero di una linea che già esiste nel programma e si termina con RETURN, la nuova linea va a sostituire la precedente che recava lo stesso numero (e in questo modo ne corregge l'errore). Se la cosa a prima vista vi confonde, provate a battere

55PRINT"QUESTA LINEA È CAMBIATA!"

Se dopo averla inserita col solito RETURN date il LIST, vedrete che la vecchia linea 55 è stata sostituita dalla nuova che avete appena impostato.

Come si cancella una linea

È pure possibile cancellare dal programma una determinata linea che non si ritiene più necessaria. Per fare questo, basta battere il numero della linea da eliminare, e terminare col RETURN. Se per esempio battete soltanto 55 e premete il tasto RETURN, al LIST vedrete che la linea 55 è stata eliminata dal programma.

Linee multi-istruzione

Finora abbiamo visto che ogni linea conteneva una sola istruzione. Il vostro computer vi consente tuttavia di avere più di una istruzione in una

stessa linea: anzi potete averne quante volete sotto uno stesso numero di linea, almeno fino a che il numero totale di caratteri non supera gli 80 (ovvero due righe sullo schermo). Ci sono alcune eccezioni, ma ne parleremo al momento opportuno. Provate a modificare la linea 10 in

```
10SCNCLR:PRINT"QUESTA È UNA LINEA MULTI-ISTRUZIONE"
```

La linea 10 contiene ora due istruzioni: il comando SCNCLR, e l'istruzione PRINT, separate da due-punti (:). Ogni volta che volete inserire più di un'istruzione, dovete separarle con i due-punti.

Sommario

Per far sì che il computer ricordi una serie di istruzioni, dobbiamo assegnare loro dei numeri di linea. In questo modo viene stabilito l'ordine con cui si vuole fare eseguire le diverse istruzioni, e al tempo stesso ci viene data la possibilità di ripetere l'esecuzione quante volte desideriamo.

Solitamente si assegnano numeri di linea con incrementi di 10, per consentire l'inserimento di linee extra in un secondo tempo, se necessario. La cosa naturalmente non è essenziale, ma risulta comoda quando capita spesso di dover fare delle aggiunte ad un programma.

Una linea di programma può esere eliminata assai semplicemente ribattendo il suo numero di linea e finendo con RETURN.

Si può visualizzare il listato del programma in memoria battendo LIST. L'esecuzione del programma viene lanciata tramite RUN.

In una stessa linea possono comparire più di una istruzione: le stesse dovranno essere separate dai due-punti (:).

Sarà bene che facciate altri esperimenti con questo programma, o che proviate addirittura a scriverne un altro vostro. In tal caso dovete battere innanzitutto NEW per eliminare dalla memoria il vecchio programma. Il tutto vi servirà a capire bene una delle parti fondamentali della programmazione in BASIC.

Variabili numeriche

Le variabili sono assai importanti per la programmazione in BASIC, e senza di esse risulterebbe in effetti veramente difficile scrivere un programma di qualche utilità pratica.

Una variabile è un valore che può essere modificato, e ve ne sono di tre tipi. Per prime esamineremo le variabili numeriche.

Per rappresentare questi valori si usano delle lettere, per esempio possiamo dire al computer che intendiamo che la lettera A rappresenti il valore 34. Provate a impostare questo breve programma (dovete prima battere NEW seguito da RETURN per liberarvi del programma attualmente in memoria):

```
10SCNCLR
20A=34:PRINT A
```

Quando date il RUN a questo programma, vedrete prima cancellare lo schermo, e poi apparire il numero 34. Infatti, con esso abbiamo detto al computer che vogliamo che la lettera A rappresenti il numero 34, e che ogni volta che ci riferiremo in seguito a tale lettera A (come ad esempio nell'istruzione PRINT A) esso deve procedere come se essa fosse il numero 34. Ovviamente potete indicare al computer di attribuire ad A un altro valore qualsiasi.

Dato che una variabile può essere impiegata per rappresentare un certo numero, possiamo operare su di esse proprio come se fossero numeri: possiamo effettuare somme, divisioni, od altri calcoli che normalmente eseguiremmo direttamente sui numeri. Provate ad aggiungere al vostro programma le seguenti linee di istruzioni:

```
30B=52:PRINTB
40PRINT"34+52=";A+B
50PRINT"34*52=";A*B
60C=B-A 70PRINT C
```

Vi forniamo alcuni chiarimenti su quello che fa il programma:

Linea 10: "pulisce" lo schermo
Linea 20: ricorda che la variabile A rappresenta il numero 34, e visualizza il numero rappresentato da A sullo schermo.
Linea 30: ricorda che la variabile B rappresenta il numero 52, poi visua-

lizza il numero rappresentato da B sullo schermo.
Linea 40: visualizza il messaggio '34+52=', poi vai a vedere cosa rappresentano le variabili A e B, fa la somma dei valori e visualizza il risultato sullo schermo, sulla medesima riga.
Linea 50: visualizza il messaggio '34*52=', poi vai a vedere cosa rappresentano le variabili A e B, fai il loro prodotto e visualizza il risultato sulla medesima riga.
Linea 60: vedi cosa rappresentano le variabili A e B, poi sottrai il valore di A da quello di B e ricorda che la variabile C rappresenta il relativo risultato.
Linea 70: vedi cosa rappresenta la variabile C, e visualizzane il valore sullo schermo.

Una variabile può essere simboleggiata con qualsiasi lettera, o combinazione di lettere e numeri, per esempio: B, SALVE, FRED, A3, ZT99 e NUMERO9 sono tutti validi nomi di variabili. Tuttavia, ogni nome di variabile *deve* iniziare con una lettera, per cui A9 è valido, mentre 9A non può essere usato.

Non si può usare un nome di variabile che contenga un comando, così p. es. non si può impiegare PRINTER come nome di variabile (rammentate che PRINT è un comando BASIC), così come non si possono impiegare PRUNO (per via di RUN) e LISTA (per via di LIST).

Un nome di variabile può essere di lunghezza qualsiasi (purchè stia in una linea di programma), ma ricordatevi che il computer le identifica e distingue solo in base alle prime due lettere del nome. Ciò significa che NUM e NUMERO equivalgono entrambe a NU, e quindi per il computer sono la stessa variabile.

Il computer assegna ad ogni variabile il valore 0 sino al momento in cui voi la impiegate, per cui potete anche scrivere A=B prima ancora di avere assegnato un valore a B: in questo caso il computer attribuirà ad A il valore 0.

Infine, finora abbiamo attribuito un valore ad una certa variabile scriven-semplicemente, do p. es., A=1. Esiste un comando BASIC detto LET che può venire usato anche per assegnare valori alle variabili. In pratica tuttavia questo comando risulta superfluo, perchè per il computer LET A=1 è letto come A=1 per assegnare un valore alla variabile A. Se trovate LET in un programma potete quindi tranquillamente tralasciarlo.

Variabili intere

Le variabili a cui ci siamo riferiti prima erano capaci di rappresentare numeri qualsiasi, inclusi quelli con una parte decimale (cosiddetti "numeri reali").Esiste un altro tipo di variabile che può invece rappresentare soltanto numeri interi (o semplicemente "interi"), ossia privi di parte decimale: questo tipo di variabile è detto "variabile intera".

Come per le variabili già citate, le variabili intere possono avere nomi che contengono una qualsiasi combinazione di lettere e numeri (purchè il primo carattere sia una lettera), ma si distinguono perchè l'ultimo carattere deve essere il simbolo del "percento" (%). Così ad esempio FRED% è un nome valido per una variabile intera, mentre FRED è il nome di una variabile normale (numerica).

Come si vede, una variabile intera ed una numerica normale possono avere lo stesso nome (a parte il %), e possono rappresentare numeri diversi (la variabile intera, naturalmente, può solo rappresentare un numero intero).

Provate ora questo breve programma, che serve a mostrare le differenze fra i due tipi di variabili (ricordatevi sempre di fare prima NEW):

```
10SCNCLR
20H=56.276:H%=56.276
30PRINTH,H%
```

Questo programma, dopo aver cancellato lo schermo, assegna il valore alla 56.276 variabile H: poi assegna alla variabile H% lo stesso valore, ma, dato che H% può accogliere solo valori interi, essa riceve il valore 56, come si può osservare quando la linea 30 procede a stampare sullo schermo i valori delle due variabili.

Avrete probabilmente notato la presenza di una virgola di separazione fra le due variabili nella linea 30. Questo simbolo fa sì che il valore di H% venga stampato a partire dalla posizione (colonna) n.o 11 dello schermo.

Variabili stringa

Un altro tipo di variabile è la "variabile stringa". Essa serve per rappresentare lettere od altri caratteri, oltre ai numeri, e si distingue dagli altri due perchè il suo nome termina col simbolo "dollaro" ($).

Dopo aver fatto NEW, provate ad impostare il seguente breve programma che utilizza delle variabili stringa:

```
10SCNCLR
20A$="SALVE":B$="A TUTTI"
30PRINT A$;" ";B$
```

Come potete vedere, i caratteri che una data variabile stringa deve rappresentare (=che ad essa vengono assegnati) devono stare racchiusi fra due virgolette.

Ci sono però alcune cose che non potete fare con le variabili. Eccone alcuni esempi.

A="FRED"	una variabile numerica non può contenere dei caratteri
PRINTER=3	il nome non deve contenere un comando
PRUNO$	BASIC
="SALVE"	così pure il nome d'una variabile stringa.
B%=A$	una variabile intera o numerica non può avere assegnato un valore "stringa", e viceversa

Sommario

Ci sono tre tipi di variabili: variabili numeriche, variabili intere e variabili stringa.

Le variabili numeriche hanno il nome costituito da una lettera o gruppo di lettere e numeri (p. es. Z, FD2, H34) e possono ricevere assegnato soltanto un valore numerico.

Le variabili intere hanno nomi costituiti allo stesso modo ma devono terminare col simbolo % (p. es. Z%, FD2%,H34%), e possono accogliere solo valori interi (ossia di numeri privi di parte decimale: se viene assegnato un numero con parte decimale, questa viene ignorata).

Le variabili stringa hanno nomi costituiti come le precedenti, ma devono terminare col simbolo $ (p. es. Z$, FD2$, H34$), e possono accogliere qualsiasi gruppo di caratteri: questi devono essere racchiusi fra virgolette.

Si può usare il comando LET per assegnare valori alle variabili, ma ciò non è strettamente necessario.

A questo punto sarà bene che sperimentiate un pò i vari tipi di variabili

prima di procedere al capitolo successivo. È importante che abbiate ben compreso come le stesse funzionano, per cui un pò di tempo speso con loro per accertare di aver ben compreso tutto potrà risparmiarvi ore di frustrazione in seguito.

LIST

Il comando LIST lo avete già utilizzato per visualizzare la serie di istruzioni di un programma sullo schermo, ma esso può risultare anche molto più utile. È posibile ad esempio LISTare soltanto parte di un programma; per esempio se battete

LIST 30-90

vedrete listate le linee da 30 a 90. Se volete esaminare il listato del programma dall'inizio sino alla linea numero 70 fate
LIST -70

ed analogamente, se volete vedere solo le ultime righe di un programma a partire - p. es. - dal numero 100 in poi, battete

LIST 100-

In altri casi potrebbe interessarvi esaminare (p. es. per modificarla) la sola linea 25: allora farete

LIST 25

È possibile rallentare la velocità con cui il programma viene listato, basta tenere premuto il tasto Commodore. Se, per un motivo qualsiasi, volete arrestare il listaggio di un programma, premete il tasto RUN/STOP

RUN

Anche il comando RUN lo avete già adoperato prima, e sapete quindi che serve per fare eseguire dall'inizio un programma. È pure possibile far partire l'esecuzione del programma da una certa linea diversa dalla prima: così, se volete che il programma cominci l'esecuzione dalla linea numero 50, batterete

RUN 50

Il comando RUN reinizializza a zero tutte le variabili numeriche, o le "svuota" di ogni carattere se si tratta di variabili stringa.

NEW

Avete già incontrato anche il comando NEW, che serve a cancellare dalla memoria del computer qualsiasi programma, ed al tempo stesso reinizializza (come RUN) tutte le variabili. Dovete quindi usare cautela con questo comando, perchè una volta eseguito NEW qualsiasi programma è perso.

INPUT

Il comando di INPUT risulta assai comodo nei programmi, perchè permette al computer di rivolgere delle domande all'utilizzatore, ed agire poi di conseguenza in base alla risposta. Quello che il comando INPUT fa in pratica è attendere che venga impostata la risposta, che poi viene assegnata ad una opportuna variabile. Eccovi un breve programma che illustra l'impiego di INPUT:

```
10SCNCLR
20PRINT"SALVE, COME TI CHIAMI ";:INPUT NOME$
30PRINT"IMPOSTA UN NUMERO QUALSIASI"O:INPUT A
40SCNCLR
50PRINT"SALVE ";NOME$
60PRINT"HAI BATTUTO IL NUMERO ";A
```

Se date il RUN a questo programma, vedrete cancellarsi lo schermo, e poi vi verrà chiesto di battere il vostro nome (notate il punto interrogativo che compare automaticamente, e segnala che il computer si attende una vostra risposta dalla tastiera alla richiesta di INPUT). Non succede nulla fino a quando non avete battuto il vostro nome seguito da RETURN ; una volta fatto questo, vi verrà richiesto di introdurre un numero, e nuovamente nulla succederà sino a che non avrete battuto questo (seguito dal solito RETURN). Una volta fatto anche questo, lo schermo si cancellerà di nuovo, ed il computer vi darà il suo saluto e vi indicherà quale numero avete impostato.

Se esaminate il programma precedente, potrete notare come il vostro nome viene memorizzato (assegnato) nella variabile stringa NOME$, mentre il numero che battete viene assegnato alla variabile numerica A. In effetti, esso dice al computer di richiedere l'introduzione (INPUT) di qualcosa dalla tastiera, e di assegnare quanto viene impostato ad una variabile. In questo caso la variabile NOME$ si vede assegnare il vostro nome, e la variabile A il numero che avete impostato.

Come certamente noterete dal programma, abbiamo usato un'istruzione

PRINT per richiedere l'informazione che si doveva battere in risposta. È possibile però incorporare un breve testo di richiesta nell'istruzione INPUT Per esempio, invece di avere due istruzioni come nella linea

```
10PRINT"QUANTO FA 27+49";:INPUT Z
```

possiamo scrivere

```
10INPUT"QUANTO FA 27+49";Z
```

Questa nuova linea farà sì che venga visualizzato il messaggio "QUANTO FA 27+49" sullo schermo, e poi si attenda il dato battuto in risposta. Tale risposta viene poi assegnata alla variabile Z.

Sommario

Il comando (istruzione) INPUT attende che venga introdotto qualcosa dalla tastiera, e ne assegna il valore ad una variabile. Nell'istruzione di INPUT si può incorporare un messagio che risparmia l'uso di un PRINT separato.

CAPITOLO 4

Programmi strutturati

IF...THEN...ELSE

Uno dei compiti più utili che un computer può svolgere è confrontare fra loro due valori. In certi programmi la cosa è essenziale, permettendo con questo mezzo al computer di svolgere diverse operazioni assai utili, come la ricerca dell'indirizzo e numero di telefono di un certo nominativo, cosa che viene svolta confrontando ciascun nome di un certo elenco con il nome specificato.

Ci sono tre comandi che, in combinazione, consentono di ordinare al computer di confrontare un certo valore con un altro: si tratta di IF, THEN ed ELSE (che significano rispettivamente SE....IN TAL CASO...ALTRIMENTI..). I tre comandi combinati costituiscono quella che vien detta una "struttura", quando vengono usati assieme come nell'esempio che segue:

```
IF Z=42 THEN K=9:ELSE K=3
```

Questo segnala al computer di verificare se la variabile Z rappresenta il numero 42, ed in tal caso (THEN) assegnare il valore 9 alla variabile K; altrimenti (ELSE) K riceve assegnato il valore 3. Se il valore di Z è 42, la parte che segue ad ELSE viene ignorata, ed il computer passa all'esecuzione della linea immediatamente successiva.

Esistono alcuni simboli che si impiegano nel confronto fra due valori. Essi sono '>' (maggiore di), '<' (minore di), ed = (eguale a): essi possono anche venire combinati, per cui >= significa "maggiore od eguale a" ("non inferiore a"), ecc. Eccone qualche esempio:

```
IF FRED = 32 THEN JOHN = 30: ELSE JOHN = 22
```

che significa: "se (IF) il valore della variabile FRED è 32 assegna alla variabile JOHN il valore 30; altrimenti (ELSE) il valore 22".

```
IF A$<>"Y" THEN SCNCLR
```

che vuol dire "se la variabile stringa A$ non rappresenta la lettera Y

cancella lo schermo. Se A$ è proprio Y, passa all'esecuzione della linea successiva".

IF KK3>99 THEN RUN

"Se il valore della variabile KK3 è maggiore di 99, esegui il RUN del programma (dall'inizio)".

Ci sono pure altri due operatori. Si tratta di AND e di OR, che vengono usati come mostrano questi esempi:

IF A=1 AND B=1 THEN C=1

"Se il valore della variabile A è 1, e contemporaneamente quello di B=1, allora assegna 1 come valore di C".

IF A=1 OR B=1 THEN C=1

"Se il valore di A=1, oppure se quello di B=1, allora assegna a C il valore 1"

Qualsiasi istruzione che segue ad un IF verrà eseguita soltanto se la condizione specificata risulta vera.

Ecco un esempio di programma che incorpora la maggior parte dei comandi che avete imparato sinora.

```
 10 SCNCLR
 20 INPUT"SALVE, CI CONOSCIAMO";IN$
 30 IFIN$="S"ORIN$="SI"THENPRINT"SPIACENTE,
     MA NON RAMMENTO IL TUO NOME."
 40 INPUT"COME TI CHIAMI";NOME$
 50 PRINT"MI PUOI CHIAMARE COMMODORE!"
 60 INPUT"TI SPIACE SE TI CHIEDO CHE ETA'
    HAI";QU$
 70 IFQU$="S"ORQU$="SI"THENPRINT"IN TAL CASO
    NON TE LA CHIEDO."
 80 IFQU$="N"ORQU$="NO"THENINPUT"DIMMI QUAN
    TI ANNI HAI";ETA$
 90 PRINT
100 PRINT"SCUSA, MA ORA DEVO ANDARE!"
110 PRINT"SPERO DI INCONTRARTI ANCORA, ";NO
    ME$
```

Ed eccovi la spiegazione del suo funzionamento, linea per linea:
Linea 10: Cancella lo schermo **Linea 20**: visualizza il messaggio "SALVE, CI CONOSCIAMO", ed attendi la risposta, che va assegnata alla variabile IN$
Linea 30: se il carattere assegnato alla variabile IN$ è 'S', ovvero sono i caratteri 'SI', visualizza il messaggio "SPIACENTE, MA NON RAMMENTO IL TUO NOME"
Linea 40: visualizza messaggio "COME TI CHIAMI" ed attendi la risposta, che va assegnata alla variabile NOME$
Linea 50: visualizza messaggio "MI PUOI CHIAMARE COMMODORE!"
Linea 60: visualizza messaggio "TI SPIACE SE TI CHIEDO CHE ETA' HAI", ed attendi la risposta, da assegnare alla variabile QU$
Linea 70: se i caratteri forniti in risposta per QU$ sono 'S' oppure 'SI' visualizza messaggio "IN TAL CASO NON TE LA CHIEDO"
LINEA 80: Se i caratteri assegnati a QU$ sono 'N' oppure visualizza messaggio "DIMMI QUANTI ANNI HAI" ed attendi la risposta, da assegnare alla variabile ETA$
Linea 90: stampa una riga vuota (interlinea)
Linea 100: visualizza messaggio "SCUSA, MA ORA DEVO ANDARE!" **Linea 110**: visualizza messaggio "SPERO DI INCONTRARTI ANCORA, ";NOME$

Sommario

La struttura IF..THEN...ELSE viene utilizzata quando si vuol confrontare un certo valore con un altro, e può essere applicata a tutti e tre i tipi di variabili, nonchè a costanti numeriche o stringa (come p. es. in IF A$="Y", o IF ZZ=22).

Alcuni operatori speciali vengono impiegati assieme a IF...THEN...ELSE nel confronto fra valori; essi sono '>' (maggiore di), '<' (minore di), ed '=' (eguale a), che possono anche essere combinati fra di loro: p. es. '>=' significa "maggiore od eguale a".

Si possono anche usare gli operatori AND ed OR, quando più di una condizione deve essere verificata prima che il computer svolga determinate azioni: come in IF A=2 AND B=1 THEN PRINT"SALVE : il computer stamperà SALVE sullo schermo soltanto se A rappresenta il numero 2 e contemporaneamente B rappresenta il valore 1.

Cicli FOR...NEXT

Spesso risulta necessario svolgere determinate istruzioni più volte di seguito. Per esempio, si può avere un programma come questo che vi mostriamo che visualizza i multipli di 15 sino a 10*15:

```
10SCNCLR
20PRINT1*15
30PRINT2*15
40PRINT3*15
50PRINT4*15
60PRINT5*15
70PRINT6*15
80PRINT7*15
90PRINT8*15
100PRINT9*15
110PRINT10*15
```

Anche se scritto in questo modo il programma funziona perfettamente, è evidente quanto risulti scomodo da impostare, e anche consumatore di memoria. Quando volete compiere ripetutamente una serie di istruzioni sono disponibili due comandi, chiamati FOR e NEXT, che servono proprio a rendere il tutto più facile. Il modo di funzionamento di questi comandi in combinazione è questo:

```
10FOR N=1 TO 10
20PRINT N*15
30NEXT N
```

Questo programma assai corto, di sole tre istruzioni svolge esattamente le stesse funzioni del primo programma con i 10 PRINT separati. Ecco come funziona:

Linea 10: indica al computer che deve cominciare a ripetere le istruzioni che stanno fra i comandi FOR e NEXT per 10 volte
Linea 20: esegue il prodotto del valore corrente di N per 15, e visualizza il risultato sullo schermo
Linea 30: Se il valore rappresentato dalla variabile N è minore di 10, incrementa il valore di N di 1, e ritorna alla linea 20.

Se provate a cambiare la linea 10 così:

```
10FOR N=20 TO 30
```

e date il RUN al programma, vedrete comparire i multipli di 15 da 20*15

sino a 30*15. Infatti in questo caso si è detto al computer di far partire il valore di N da 20 e di incrementarlo successivamente ad ogni passaggio nel ciclo sino a 30.

Un altro comando, STEP, aggiunto alla fine dell'istruzione FOR e seguito da un numero, permette di modificare il valore di cui N (od altra variabile simile usata dopo FOR) viene incrementato ad ogni passaggio nel ciclo. Per esempio, la linea 10 può venire modificata così:

```
10FOR N=20 TO 30 STEP 2
```

e date il RUN, vedrete comparire sullo schermo i multipli pari di 15 da 20*15 a 30*15.

Potete anche, con un valore negativo posto dopo STEP, procedere a ritroso, ossia decrementare di un valore fisso N ad ogni ciclo. Modificate la linea 10 in

```
10FOR N=30 TO 20 STEP —1
```

al RUN vedrete comparire successivamente i multipli di 15, partendo da 30*15 sino a 20*15.

I cicli tipo FOR...NEXT possono anche essere "inseriti" (taluno dice "annidati") uno nell'altro. Provate questo esempio:

```
10FOR N=1 TO 10
20FOR M=1 TO 500:NEXT M
30PRINT N 40NEXT N
```

Quando date il RUN al programma vedrete comparire, ma con lentezza, i numeri da 1 a 10 in successione sullo schermo. Nella linea 20 abbiamo infatti creato un altro ciclo, "inserito" nel primo, che serve in pratica solo a girare su se stesso senza fare niente di particolare (tranne che perdere tempo...). In tal modo l'esecuzione del programma viene rallentata, ed il primo ciclo viene eseguito più lentamente. Eco un altro esempio di programma con più cicli "annidati" uno entro l'altro:

```
10 SCNCLR
20 FORN=1TO12
30 PRINT"LA TABELLINA DEL";N:PRINT
40 FORM=1TO12
50 IFM<10THENPRINT"  ";
```

```
 60 PRINTM;"*";N;"=";M*N
 70 NEXTM
 80 FORZ=1TO2000:NEXTZ
 90 SCNCLR
100 NEXTN
110 RUN
```

Questo programma contiene due cicli inseriti in uno più esterno: il primo dalla linea 40 alla 70, l'altro alla linea 80. Il ciclo principale, che va dalla linea 20 alla 100, serve a selezionare la "tabellina" desiderata. I multipli partono da 1 per arrivare, con incrementi della variabile N di 1, sino a 12.

Il primo dei cicli "inseriti" in quello più esterno usa la variabile M, che varia anch'essa da 1 sino a 12. La linea 50 serve a visualizzare uno spazio se il valore di M è minore di 10, in modo da allineare i numeri in colonna, con migliore estetica: potete vedere esattamente come questa linea funziona se provate ad eliminarla dal programma e date RUN. Il valore di M viene moltiplicato successivamente per quello (al momento) costante di N, così da visualizzare tutti i multipli di N sino a 12*N.

Il secondo ciclo "inserito" che usa la variabile Z è ancora una volta un "ciclo di ritardo", che conta sino a 2000 per dare il tempo di leggere quanto viene stampato sullo schermo.

Lo schermo viene infatti ripulito ogni volta tramite SCNCLR (linea 90). Una volta che il programma è terminato, esso ricomincia automaticamente, per via del RUN posto nela linea 110.

Sommario

I cicli FOR...NEXT servono per eseguire ripetutamente un certo gruppo di istruzioni. Il valore della variabile usata nel ciclo (cosidetta variabile "di controllo" del ciclo, come la N) viene normalmente incrementata di 1 ad ogni passaggio attraverso il ciclo.

Con l'uso del comando STEP posto alla fine dell'istruzione FOR e seguito da un numero si può modificare il valore di tale incremento. Così ad es. STEP 3 incrementa il valore della variabile di controllo di 3 ogni volta; mentre STEP -2 decrementa la variabile di 2 ad ogni ciclo.

L'istruzione NEXT segnala in ogni caso dove il ciclo termina. A questa può essere aggiunto il nome della variabile (ad es. NEXT N), ma la cosa

non è essenziale, salvo in presenza di più cicli che possono dare luogo a confusioni.

GOTO

Come ormai dovreste sapere, quando il computer esegue un certo programma esso lo percorre seguendo l'ordine dei numeri di linea. E' possibile tuttavia modificare questo ordine di esecuzione, e fare in modo che, per fare un esempio, il computer inizi coll'eseguire la linea 10, poi passi ad eseguire le linee da 100 a 150, poi ritorni alla linea 20 e prosegua da qui in avanti.

L'istruzione GOTO segnala al computer che deve saltare (GOTO) ad un certo numero di linea,che viene specificato dopo tale comando, e proseguire da lì l'esecuzione del programma. Provate con questo programmino:

```
10PRINT"#";
20GOTO 10
```

Se date il RUN a questo programma vedrete che il simbolo # viene stampato, senza interruzione, riga dopo riga sullo schermo. Per fermare questo programma, che non terminerebbe mai, dovete premere il tasto RUN/STOP.

Il motivo per cui il programma continua senza fermarsi mai sta nel fatto che, dopo che il primo simbolo # è stato stampato, dal PRINT della linea 10, la linea 20 successiva dice al computer di tornare alla linea 10 stessa, per cui viene stampato di seguito (notare il ;) un altro #, e così via senza requie.

GOSUB e RETURN

L'istruzione (comando) GOSUB è molto simile a GOTO. Essa ordina al computer di saltare (GO) ad una certa SUBroutine (sottoprogramma) e di continuare da lì l'esecuzione. Quando poi il computer raggiunge un'istruzione RETURN (che deve sempre essere presente in una subroutine), il computer passerà ad eseguire l'istruzione immediatamente successiva al GOSUB di partenza.

Una subroutine o sottoprogramma è "un programma entro un programma", ossia un breve pezzo di programma che serve per svolgere un compito particolare, e può essere impiegato più volte nel corso del pro-

gramma principale. La posibilità di GOSUB a queste subroutine serve ad evitare di dover riscrivere ogni volta il medesimo pezzo di programma, che verrà impostato invece una sola volta.

Ecco un breve programma che illustra l'uso del GOSUB:

```
10PRINT "ROUTINE 1"
20PRINT"**********"
30GOSUB 50
40GOTO 10
50PRINT"ROUTINE 2"
60PRINT"##########
70RETURN
```

Quando si dà il RUN al programma, vedrete comparire sullo schermo:

```
ROUTINE 1
**********
ROUTINE 2
##########
ROUTINE 1
**********
ROUTINE 2
##########.......
```

ripetendo il tutto più e più volte senza fermarsi, almeno sino a che non premete RUN/STOP.

Ecco come funziona il programma:

Linea 10: visualizza il messaggio "Routine 1"
Linea 20: visualizza 10 asterischi **********
Linea 30: salta alla subroutine che inizia dalla linea 50
Linea 40: torna alla linea 10 e riprende da lì
Linea 50: visualizza il messaggio "Routine 2"
Linea 60: visualizza ##########
Linea 70: ritorna all'istruzione immediatamente successiva a quella dove era il GOSUB (in questo caso, l'istruzione 40 che segue alla 30 dove sta il GOSUB)

Vi sarete ormai accorti che se in un programma compare una linea come questa:

100GOTO 50:PRINT"SALVE"

l'istruzione PRINT non verrà mai eseguita, perchè non appena raggiunge l'istruzione GOTO il computer salta alla linea 50 e prosegue da quel punto il programma. Se però la linea 100 è scritta così:

100GOSUB 50: PRINT"SALVE"

l'istruzione PRINT verrà eseguita quando, trovato il RETURN alla fine subroutine da linea 50 che gli dice di "rientrare", il computer tornerà all'istruzione immediatamente successiva al GOSUB, che in questo caso è la PRINT.

Una delle cose che non potete fare tramite le istruzioni GOTO o GOSUB è di sostituire al numero di linea che li segue una variabile: se per esempio volete che si esegua GOTO 10, non potete sostituire il 10 con una variabile, se rappresenta proprio il numero 10.

GOTO e GOSUB possono naturalmente essere impiegati anche in struttureIF..THEN..ELSE, e la cosa può risultare molto utile.

Sommario

Il comando GOTO dice al computer di saltare ad un certo numero di linea e di proseguire da quel punto il programma.

Il comando GOSUB dice al computer di saltare ad una subroutine che inizia da un certo numero di linea e di proseguire da lì il programma. Quando però trova un comando RETURN, il computer dovrà tornare all'istruzione posta immediatamente dopo quella in cui era il GOSUB.

Ogni istruzione posta dopo un GOSUB verrà quindi regolarmente eseguita dopo il RETURN.

Il numero posto dopo GOTO o GOSUB non può essere sostituito da una variabile.

Come correggere i programmi

Finora, ogni volta che commettevate un errore di battitura nei vostri

programmi, dovevate ribattere completamente la linea sbagliata. È però possibile correggere, o modificare, i programmi senza dover ricorrere a questo scomodo rimedio. Battete questo programmino esattamente come lo vedete scritto:

```
10SCMCLR
20PRIINT"SALVEE"
30GTO 20
```

Ovviamente il programma presenta molti errori, ma sarebbe certamente fastidioso doverlo ribattere tutto da capo. Fortunatamente, questo non è necessario, dato che possiamo "editare" (come si suol dire) il programma.

Per prima cosa, spostate il cursore (con i tasti-freccia) sulla M di SCMCLR, e poi battete N seguito da RETURN. La linea 10 sarà così corretta.

Il prossimo errore lo troviamo alla linea 20, dove anzi ce ne sono due. Per correggerli, dovete spostare il cursore sulla N di PRIINT (che precede la I di troppo), e quindi premere il tasto INST/DEL. Poi dovete muovere il cursore sopra la seconda E in fine linea, ed ancora premere INST/DEL. Premendo RETURN vedrete che la linea 20 adesso è corretta.

Nel comando GOTO di linea 30, infine, manca una O. Per inserire la lettera mancante dovete spostare il cursore sino sopra la T, quindi premere SHIFT ed INST/DEL assieme. Vedrete comparire uno spazio, pronto per accogliere la O che batterete. Premendo RETURN anche la linea 30 apparirà corretta.

Se, mentre state modificando una certa linea, cambiate idea e decidete di lasciarla com'è, potete annullare le modifiche sino allora apportate o premendo SHIFT CLEAR/HOME (che provoca la cancellazione dello schermo), oppure premendo SHIFT e RETURN. Potete anche semplicemente spostare il cursore fuori della linea in questione, perchè le modifiche non diventano effettive sino a quando non avete premuto il tasto RETURN.

Salvataggio dei programmi su cassetta

Se volete riutilizzare il programma che avete impostato in precedenza, è certamente poco comodo ribatterlo ogni volta da capo. Per ovviare a ciò potete memorizzare (o "salvare", come si usa dire) il programma su di una normale cassetta audio.

Probabilmente sarete già in possesso di un registratore Commodore Datassette 1531. Se così non fosse, sarebbe bene ne acquistaste uno, perchè anche se doveste usare un'unità disco, molti programmi commerciali sono disponibili soltanto su cassetta. Il Datassette va collegato nella presa marcata CASSETTE sul retro del computer (prima di farlo, spegnete il vostro computer). Ora vi serve qualche nastro vergine: ogni nastro di buona marca va bene, purchè siano indicati per "normal bias": però esistono anche nastri in cassetta appositamente fabbricati per l'uso con i computer.

Una volta effettuati tutti i collegamenti, e con una cassetta idonea inserita nel registratore, riavvolgete quest'ultima sino a vedere comparire il tratto non magnetizzato (generalmente colorato) di fronte alla testina. Siete così pronti a salvare il vostro programma. Proveremo per esercizio a salvare questo programma:

```
10SCNCLR
20PRINT"QUESTO PROGRAMMA È STATO SALVATO SU"
30PRINT"NASTRO E REGOLARMENTE RICARICATO"
40FOR N=1 TO 25
50PRINT"**************************************"
60NEXT N
70FOR N=1 TO 2000:NEXT N 80RUN
```

Ed ora tutto quel che resta da fare è di battere

SAVE"PROGRAMMA"

Vedrete comparire un messaggio che dice PRESS PLAY AND RECORD ON TAPE ("premi i due tasti PLAY e RECORD assieme"), cosa che farete, esattamente come quando volete registrare qualcosa su di un normale registratore a cassette. Vedrete che lo schermo diventa bianco, ed il motore del registratore si avvierà automaticamente. Dopo un certo tempo lo schermo tornerà normale ed il nastro si fermerà: il vostro programma è stato registrato sulla cassetta.

Il comando SAVE"PROGRAMMA" dice al computer di riversare una copia del programma correntemente in memoria sul nastro, e di assegnargli il nome PROGRAMMA (l'assegnare un nome ad un programma ne rende più facile in seguito la ricerca). Come vedete, il nome va posto fra virgolette.

Verifica del programma registrato

Potete accertarvi che il programma sia stato salvato correttamente usando l'apposito comando VERIFY. Riavvolgete la cassetta sino all'origine, e battete

VERIFY"PROGRAMMA"

Il computer vi segnalerà sullo schermo PRESS PLAY ON TAPE ("premi il tasto PLAY di riascolto"), e non appena avrete eseguito lo schermo diventerà bianco ed il motore si avvierà. Il computer sta cercando sul nastro il programma che avete salvato con il nome PROGRAMMA. Quando lo trova, visualizzerà il messaggio

FOUND PROGRAMMA

VERIFYING

e quindi si arresterà. A guadagno di tempo, potete a questo punto premere il tasto Commodore, che dice al computer di non attendere. Il computer procederà allora a confrontare il programma che è sul nastro con quello che è ancora in memoria. Se essi risultano identici (come dovrebbe essere, dato che quello in memoria è proprio il programma che avete salvato sulla cassetta) lo schermo ad un certo punto tornerà normale ed il computer vi informerà che tutto è OK. Se il programma non fosse stato salvato correttamente, invece, apparirà il messaggio

VERIFY ERROR

Se dovesse capitare, riavvolgete il nastro, e riprovate da capo. Se continua a comparire il messaggio che dice che il salvataggio non è riuscito correttamente, provate con un altro tipo di nastro: e se, dopo aver sperimentato vari nastri, non avete ancora successo, dovrete portare il computer ed il Datassette al vostro rivenditore Commodore per farli controllare e, se è il caso riparare.

Caricamento di un programma da nastro

Spero comunque che il vostro salvataggio sia pienamente riuscito ed abbiate il vostro programma felicemente registrato sulla cassetta. Ora battete NEW e poi

LOAD"PROGRAMMA"

Il computer vi segnalerà PRESS PLAY ON TAPE (come per la verifica), e non appena avrete seguito il consiglio lo schermo al solito passerà al bianco ed il motore partirà: il computer ricerca sul nastro un programma che abbia il nome PROGRAMMA come desiderato. Non appena lo trova visualizzerà

FOUND PROGRAMMA

LOADING

Ci sarà ancora una pausa, che può essere saltata premendo il tasto Commodore. Lo schermo tornerà bianco mentre il computer procede al caricamento del programma. Una volta comparso il messaggio di OK potete fare LIST o RUN, e vedrete che il vostro programma è in memoria.

Il nome che segue i comandi SAVE, VERIFY e LOAD non è obbligatorio, anche se è certamente una buona idea assegnare un nome diverso a ciascun programma. Per esempio, se battete solo LOAD e premete RETURN, il computer caricherà in memoria il primo programma che incontra sul nastro.

Alla fine dell'istruzione LOAD o SAVE potete porre due numeri. Essi indicano rispettivamente il numero della periferica, ed il flag di rilocazione. Per esempio, il comando

SAVE"GRAFICO",8,1

salva il programma in memoria su disco, con numero di periferica 8, dandogli il nome GRAFICO. Per una spiegazione del significato dei vari numeri assegnati alle periferiche, vogliate consultare il vostro "Manuale d'uso". L'1 aggiunto in fondo, poi, dice al computer di aggiungere un marcatore di EOT (End OF Tape = fine nastro) alla fine del programma. Il che sigifica che quando il computer ricerca un programma su di un nastro, e giunge ad uno di questi marcatori EOT, per lui è come se fosse arrivato alla fine del nastro, per cui viene fornito un messaggio FILE NOT FOUND ERROR.
Analogamente, il comando

LOAD"GRAFICO",1,1

fa eseguire al computer il caricamento dalla cassetta del programma di nome GRAFICO (il primo '1' dice al computer di effettuare il caricamento da nastro), e di collocarlo in memoria a partire dalla locazione da cui era stato prelevato. L'1 alla fine del comando citato serve appunto a far

ricaricare il programma nella stessa area di memoria da cui era stato prelevato. Se questo parametro viene tralasciato, il computer caricherà il programma a partire dall'inizio dell'area BASIC. In effetti, questo tipo di comando lo impiegherete esclusivamente per caricare programmi in linguaggio macchina, come spiegheremo più oltre, dato che questi spesso sono posti in zone diverse della memoria.

Sommario

Il comando SAVE viene usato per fare una copia del programma in memoria, registrandolo su cassetta. Al programma può essere assegnato un nome, per renderne più facile il reperimento in seguito.

Il nome del programma può essere formato da caratteri qualsiasi, per una lunghezza massima di 16 caratteri. Il nome va incluso fra virgolette.

Il comando VERIFY serve a controllare che il programma salvato corrisponde esattamente a quello in memoria. Può anche essere usato a volte per trovare dove termina sul nastro un dato programma, battendo VERIFY"nome programma" ed attendendo sino a che il computer emette un messaggio di errore.

Il comando LOAD serve a ricaricare nella memoria del computer un programma registrato su nastro. Si può specificare il nome del programma cercato, ed in tal caso sarà questo ad essere caricato; ma il nome non risulta essenziale se si sa esattamente dove comincia un dato programma sul nastro.

Certi programmi in linguaggio macchina richiedono di essere caricati con il comando LOAD"nome programma",1,1.

Altre idee

INT

Il comando INT serve per dare l'intero (arrotondato per difetto) di un dato numero. Per esempio, battendo

PRINT INT(32.3)

vedrete comparire sullo schermo il valore 32, ossia l'intero arrotondato per difetto che deriva da 32.3. L'arrotondamento è sempre verso il basso, per cui 98.1 dà 98, -54.87 dà -55, e così via.

RND

Il comando RND, come INT, è in effetti una "funzione", ossia un comando che prende un numero (l'"argomento", normalmente indicato fra parentesi) e ne ricava un valore che fornisce come risultato.

La funzione RND serve a scegliere un numero a caso fra 0 ed 1. Il modo con cui questo numero viene ricavato dipende dalla "base" da cui partiamo, e che è il numero, racchiuso fra parentesi, che segue RND, e può essere un numero negativo, positivo, o zero.

Se usiamo una base positiva, come in

PRINT RND(1)

il computer ricaverà un numero a caso (od almeno quel che più si avvicina ad un vero numero a caso, o "random" come talvolta si dice: infatti dopo un gran numero di replicazioni di RND i numeri si ripetono).

Una base negativa, come in

PRINT RND(—1)

reinizializza la base del generatore di numeri casuali. Quando si sceglie un numero a caso esso viene ricavato da una lunga serie di numeri generati in

un modo particolare: ad ogni applicazione di RND con la stessa base, viene scelto il numero successivo della serie. L'uso di una base negativa altera la posizione nella lista da cui il numero è ricavato. Nell'esempio più sopra avreste in risposta 2.99196472E-08, ma se aveste usato un'altra base negativa, come p. es. -2, avreste ottenuto un altro risultato.

Se infine si usa per base lo 0, come in

PRINT RND(0)

il numero a caso viene ricavato a seconda dello stato del "clock" (temporizzatore) interno. Questo metodo per generare numeri a caso è probabilmente il migliore possibile per i vari casi pratici, dato che è virtualmente impossibile che la serie di numeri si ripeta.

Il programma che segue sceglie 30 numeri a caso fra 1 e 10, dieci per ogni tipo di selezione di numero casuale indicato:

```
 10 SCNCLR
 20 PRINT"BASE POSITIVA":X=1
 30 GOSUB80
 40 PRINT"BASE NEGATIVA":X=-1
 50 GOSUB80
 60 PRINT"BASE NULLA":X=0
 70 GOSUB80:END
 80 FORN=1TO10
 90 PRINTINT(RND(X)*9)+1;
100 NEXTN
110 FORN=1TO4000:NEXTN
120 SCNCLR:RETURN
```

Noterete come il numero casuale selezionato nella linea 90 viene moltiplicato per 9 e quindi arrotondato. Il tutto equivale a scegliere un numero intero a caso fra 0 e 9. L'aggiunta di 1 produce un numero a caso fra 1 e 10.

CHAR

Il comando CHAR assomiglia a PRINT, in quanto serve per visualizzare dei caratteri sullo schermo. CHAR diferisce da PRINT, però, in quanto ci consente di scegliere esattamente il punto dove i caratteri saranno visualizzati sullo schermo.

Avrete probabilmente già osservato che sullo schermo si possono avere sino a 40 colonne e sino a 25 righe. Il che vuol dire che su di uno schermo possono comparire sino ad un massimo di 1000 caratteri.

Ora, provate ad immaginare una "griglia" o reticolo tracciato sullo schermo, suddivisa appunto in 25 righe e 4o colonne, che individua una sorta di quadrettatura. Supponete scritti sopra la prima riga di quadretti i numeri da 0 a 39, ed a sinistra della prima colonna i numeri da 0 a 24. Si possono così stabilire le "coordinate" di un qualsiasi quadretto, ossia di ogni "posizione di stampa", che sono il numero d'ordine della riga e della colonna a cui il quadretto appartiene. Così, se una certa lettera è stampata nell'angolino sinistro in alto dello schermo, le sue coordinate sono 0,0 (colonna 0 e riga 0). Se invece la lettera compare nell'angolino destro in basso, le sue coordinate sono 39,24 (colonna 39 della riga 24: notare l'ordine).

Ed ora che sapete come si può immaginare suddiviso lo schermo e definire la posizione di ogni carattere che vi compaia, potete cominciare ad usare CHAR. Tutto quello che dovete fare è specificare al computer in quale colonna e riga volete che compaia il primo dei caratteri del vostro messaggio, e quindi dirgli cosa deve visualizzare da lì. Provate con questo programma esemplificativo:

```
10 SCNCLR:INPUT X "COORDINATA X";
20 INPUT Y "COORDINATA Y";
30 CHAR1,X,Y,"SALVE"
40 FORN=1TO4000:NEXTN
50 RUN
```

Se date il RUN al programma, vi verrà chiesta la coordinata X, in altre parole il numero di colonna da cui il testo deve iniziare. Poi vi verrà chiesta la coordinata Y, ossia il numero di riga in cui deve cominciare il testo. Forse la cosa vi sembrerà in principio un po' difficoltosa, ma vedrete che, sperimentando un pò con diversi valori di X ed Y, ne verrete presto a capo. Una volta fatto quanto detto, vedrete comparire SALVE sullo schermo, partendo dal quadretto che avete appena scelto.

Se esaminate la linea d'istruzioni 30 del programma vedrete che vi compare il comando CHAR. Dopo di esso compaiono i valori di X e di Y, ma come potete notare c'è pure un 1 prima di questi, subito dopo CHAR. Questo valore indica al computer che volete che il testo compaia nel colore corrente. Se cambiate in 0 questo valore, il testo apparirà nel colore dello sfondo, per cui rimarrà... invisibile.

Come vedete, l'1, la coordinata X e la Y sono tutte separate da virgole. Il testo può essere di lunghezza qualsiasi, naturalmente purchè la lunghezza complessiva dell'istruzione che lo contiene non superi i soliti 80 caratteri.

Se provate a modificare la linea 30 così:

```
30CHAR1,X,Y,"SALVE",1
```

e date il RUN, vedrete che la parola SALVE compare a caratteri invertiti: ciò dipende dall'1 posto in fondo, dopo il testo. Se non si aggiunge nulla, oppure uno 0, i caratteri appariranno normali.

Sommario

Lo schermo si può immaginare suddiviso in 1000 quadretti occupabili da un carattere, su 25 righe e 40 colonne. A ciascuno di questi quadretti si possono quindi assegnare delle coordinate.

Il comando CHAR consente di scegliere la posizione da dove vogliamo venga visualizzato un certo messaggio sullo schermo.

Il testo va incluso fra virgolette, esattamente come per il comando PRINT.

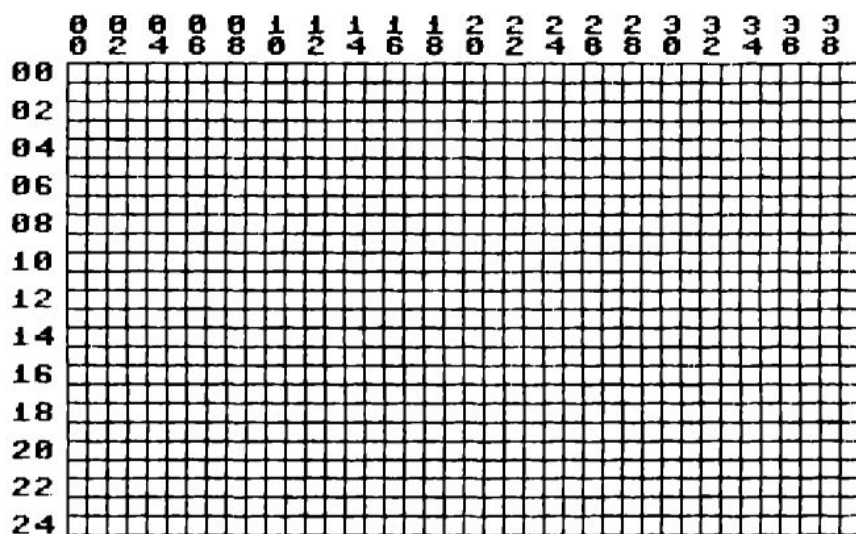
L'aggiungere '1' dopo il testo lo fa comparire a caratteri invertiti.

I caratteri possono apparire nel normale colore dei testi (CHAR 1), oppure nel colore dello sfondo (CHAR 0).

Il reticolo di CHAR

Nella figura che segue vi mostriamo come apparirebbe sullo schermo la "griglia" relativa a CHAR, con le coordinate di ciascun quadretto per i caratteri. Il quadretto posto nella 4.a riga dall'alto e nella 10.a colonna da sinistra avrebbe le coordinate 9,3.

Come usare i joystick

I cosiddetti "joystick" sono dei dispositivi versatili che possono essere usati per molti scopi diversi, anche se il più ovvio, e più comune, resta quello come comandi di controllo per i giochi. Potete collegare sino a due

joystick al vostro computer. Essi andranno inseriti nelle apposite prese marcate JOY0 e JOY1.

La lettura dello stato di un joystick è facile: si impiega a tale scopo la funzione JOY, che fornisce in risposta un valore che dipende dalla posizione del joystick:

Quindi, se il joystick che si vuol "leggere" è rivolto verso l'alto, la funzione JOY ritornerà il valore 1: se è rivolto in basso il valore 4, e così via. La pressione del pulsante di "sparo" ("fire") aggiunge 128 al valore corrispondente alla sola posizione.

Ora che sapete i valori che corrispondono alle varie posizioni del joystick, può essere utile vedere un esempio di "lettura" dello stato dei joystick. Provate dopo aver battuto il seguente programma:

```
10 SCNCLR
20 PRINT"JOYSTICK DESTRO JOYSTICK SINISTRO
   "
30 RI=JOY(1):LE=JOY(2)
40 CHAR1,5,3,RI:CHAR1,25,3,LE
50 GOTO30
```

Il joystick di destra viene letto con la funzione JOY(1), e quello di sinistra con JOY(2) (peccato che sorga un pò di confusione con le denominazioni delle relative prese, che come detto sono marcate JOY0 e JOY1).

Il programma legge il valore corrispondente al joystick destro e lo assegna alla variabile RI, mentre assegna a LE il valore del joystick sinistro. Questi due valori vengono quindi visualizzati sullo schermo. Se è pure premuto il pulsante "fire" il valore viene aumentato di 128, come potrete verificare provando a premere tale pulsante. Quando il joystick è in posizione neutra (centrale) il valore letto è 0.

GET

Uno dei modi con cui il computer può richiedere informazioni tramite la tastiera lo conoscete già: è il comando INPUT. Ma vi immaginate come si potrebbe svolgere un gioco in cui si dovesse continuamente premere il tasto RETURN dopo ogni INPUT, se si potesse usare solo questa istruzione?

Per fortuna, disponiamo di due altri tipi di comandi che si possono utilizzare per verificare se è stato premuto qualche tasto. Essi sono GET ed GETKEY. Provate questo programmino:

```
100 SCNCLR
110 PRINT"VUOI RIPETERE (S/N)?";
120 GETA$:IFA$=""THENGOTO120
130 IFA$="N"THENEND
140 IFA$="S"THENRUN
150 GOTO120
```

Se date il RUN al programma, vi verrà chiesto "VUOI RIPETERE

(S/N)?" e l'esecuzione si fermerà sino a quando avrete premuto il tasto S o N. Se premete N il programma si arresterà definitivamente, mentre se premete S il programma ripartirà da capo. La pressione di ogni altro tasto non avrà alcun effetto.

Nella linea 120 troviamo il comando GET. Come potete vedere, assieme a GET bisogna specificare una variabile (qui A$): infatti, il comando GET procede a sondare la tastiera e, se risulta premuto un tasto, il carattere corrispondente viene assegnato alla variabile specificata. Se non è premuto alcun tasto la variabile resta "vuota". Nella stessa linea 120 si verifica se A$ risulta vuota, ed in tal caso si torna alla linea 120 sino a quando un tasto non viene premuto.

Nelle linee 130 e 140 si esamina quale tasto è stato premuto. Se non sono stati premuti i tasti S o N non si ha alcuna azione, e la linea 150 ordina al computer di tornare ancora alla linea 120, in attesa del tasto giusto.

Il comando GET ha il grande vantaggio di non fermare il programma, come fa invece INPUT, però accetta solo un carattere per volta. Il programma seguita sia che sia premuto un tasto oppure no.

GETKEY

Il comando GETKEY è simile a GET, però qui si attende sino a che un tasto viene premuto, ed il programma non continua sino a che ciò non è avvenuto. Si risparmia così la verifica se la variabile specificata è vuota oppure no, ma si ha lo svantaggio che il programma si arresta. Se usiamo GETKEY possiamo modificare la linea 120 in

```
120 GETKEY A$
```

Il programma funziona ancora allo stesso modo, ma non c'è più bisogno di controllare se A$ rappresenta qualcosa, perchè GETKEY attende comunque che venga premuto un tasto prima di assegnare il relativo carattere alla variabile A$.

Sommario

Il comando GET serve a sondare la tastiera, ed assegna il carattere corrispondente a qualsiasi tasto risulti premuto ad una variabile stringa. Se non risulta premuto alcun tasto, la stringa rimane vuota.

Il comando GET, a differenza di INPUT, a cui assomiglia, non ferma il programma, e non richiede la pressione di RETURN.

Il comando GETKEY è simile a GET, ma ferma l'esecuzione del programma sino a quando non viene premuto un tasto.

I cicli DO...LOOP

Abbiamo già incontrato un tipo di ciclo iterativo, quello formato da FOR...NEXT. Esiste anche un altro tipo di ciclo, determinato dalle istruzioni DO...LOOP ("esegui" e "ricicla"). In certo modo, questo tipo di ciclo è più facile da capire del FOR...NEXT, però in genere si usa meno. Eccovi un esempio:

```
10 DO
20 PRINT "SALVE"
30 LOOP
```

Se date RUN, il computer visualizzerà a ripetizione la parola SALVE sino a quando non lo fermate premendo RUN/STOP. Ciò è motivato dal fatto che abbiamo ordinato in tal modo al computer di stampare SALVE sullo schermo, e poi di riciclare (LOOP) all'istruzione che segue immediatamente il comando DO (che nel nostro caso è ancora l'istruzione di PRINT), e ricominciare.

Può sembrare che la cosa non abbia alcuna utilità pratica. Tuttavia, se modificate il programma a questo modo la penserete diversamente:

```
10 DO UNTIL A=10
15 A=A+1
```

Se ora date il RUN, vedrete che il computer visualizza SALVE per 10 volte, e poi si ferma. Questa volta abbiamo detto al computer di eseguire quanto si trova fra DO e LOOP sino a che (UNTIL) il valore di A raggiunge 10. Dato che il valore di A parte da 0 (non c'è alcuna assegnazione iniziale) e viene incrementato di 1 ad ogni ciclo, la parola SALVE viene stampata solo per 10 volte. Se modificate ulteriormente la linea 10 in

```
10 DO UNTIL A=20
```

vedrete stampare SALVE per 20 volte.

Un'altra forma del ciclo DO...LOOP è del tipo DO WHILE... LOOP, che è molto simile a DO UNTIL...LOOP, come potrete verificare se cambiate ancora la linea 10 così:

10 DO WHILE A<10

Dando il RUN al programma si vedrà stampare SALVE per 10 volte, proprio come prima. Questa volta il computer esegue quanto sta fra DO e LOOP fintanto che (WHILE) il valore della variabile A si mantiene inferiore a 10. Non appena A raggiunge 10 il computer esce dal ciclo.

I termini UNTIL e WHILE si possono porre immediatamente dopo il DO, come sopra, od anche dopo il LOOP, per cui si potrebbero cambiare le linee 10 e 30 in

10 DO 30 LOOP WHILE A<10

ed il programma funziona esattamente allo stesso modo.

Talvolta può essere necessario uscire "prematuramente" da un ciclo DO.-..LOOP: a questo scopo si usa il comando EXIT. Provate ad aggiungere questa linea al solito programmino:

22 GET A$:IF A$="A"THEN EXIT

Ora potete uscire (EXIT) dal ciclo premendo il tasto A. Infatti con tale linea si è detto al computer di sondare la tastiera, e di eseguire EXIT se risulta premuto il tasto A.

È possibile "annidare" uno entro l'altro più cicli DO...LOOP proprio come succedeva per i cicli FOR...NEXT. Eccovi un esempio di più DO.-..LOOP inseriti uno entro l'altro:

```
10 SCNCLR
20 DOUNTILY=25
30 DOUNTILX=40
40 CHAR1,X,Y,"*"
50 FORN=1TO50:NEXTN
60 CHAR1,X,Y," "
70 X=X+1:LOOP
80 X=0:Y=Y+1:LOOP
```

Ecco come funziona questo programma:

Linea 10: solita cancellazione iniziale dello schermo
Linea 20: esegui le istruzioni poste fra DO e LOOP sino a quando il valore della variabile Y raggiunge 25
Linea 30: esegui le istruzioni poste fra i (secondi) DO e LOOP sino a quando il valore della variabile X raggiunge 40
Linea 40: visualizza un asterisco nella riga Y, colonna X
Linea 50: questo è un ciclo di ritardo
Linea 60: visualizza uno spazio nella riga Y, colonna X
Linea 70: aggiungi 1 alla variabile X, e ricicla all'istruzione posta dopo l'ultimo DO
Linea 80: assegna il valore 0 alla variabile X, ed aggiungi 1 alla variabile Y, quindi ricicla a dopo il primo DO

Il ciclo interno viene eseguito prima di quello più esterno, 40 volte per ogni passaggio nel ciclo esterno. Il ciclo interno controlla la posizione orizzontale (colonna) dell'asterisco, per cui l'esecuzione ripetuta 40 volte produce il movimento dell'asterisco lungo una riga dello schermo, di un passo per volta. Ogni volta che il ciclo interno ha compiuto 40 passi l'asterisco viene abbassato di una riga, sino a quando (dopo 25 righe) raggiunge il fondo dello schermo.

Sommario

Le istruzioni poste fra i comandi DO e LOOP in un programma vengono eseguite ripetutamente sino a quando (UNTIL) una certa condizione viene soddisfatta, oppure fintanto che (WHILE) una certa condizione resta soddisfatta.

I comandi UNTIL e WHILE si possono disporre subito dopo DO, oppure dopo LOOP.

Il comando EXIT permette di uscire da un ciclo, sia che questo sia stato completato oppure no.

Più cicli DO...LOOP si possono inserire l'uno entro l'altro, proprio come nel caso dei cicli FOR...NEXT.

DIM e le variabili multiple

Una variabile multipla è una nuova forma di variabile leggermente più

complicata di quelle che abbiamo finora viste. Le variabili multiple risultano più utili in quanto sono caratterizzate da un numero indice.

La prima volta possono risultare un pò difficili da capire, comunque un modo pratico per intenderle è questo. Immaginate di avere un libro dotato di 12 pagine, numerate da 0 ad 11. Su ciascuna pagina sta scritto un numero. Questo libro può illustrare il concetto di variabile multipla. Se per esempio vogliamo vedere il numero che è scritto sulla pagina 5 prenderemo il libro, lo apriremo alla pagina 5 e leggeremo il suo contenuto.

Una variabile multipla (o "vettore", come viene spesso anche chiamata) assomiglia a questo libro con le pagine numerate. Essa funziona in modo analogo: se per es. supponiamo di avere una variabile multipla di nome LIBRO, e vogliamo vedere cosa contiene il quinto numero rappresentato da LIBRO imposteremo

PRINT LIBRO(5)

Il computer esaminerà i contenuti della variabile multipla LIBRO e vedrà quale è il valore del suo quinto "elemento", visualizzandolo poi sullo schermo.

Purtroppo, le variabili multiple "consumano" molta memoria del computer, per cui è importante specificare sempre quanti e quali vettori vogliamo definire, e quanti elementi vogliamo memorizzare in essi. Per fare questo si impiega l'istruzione DIM. Per esempio, se si vogliono memorizzare 14 numeri in un vettore chiamato LIBRO useremo un'istruzione del tipo

10DIM LIBRO(13)

col che si dice al computer di riservare abbastanza spazio di memoria da contenere 14 numeri (l'indice o numero d'ordine varierà da 0 a 13, ecco spiegato il 13 anzichè 14). Possiamo ora aggiungere le linee che seguono per avere un programma che vi richiede i valori di 14 numeri e poi li stampa sullo schermo:

```
20 SCNCLR
30 FORN=0TO13
40 INPUT"PER FAVORE IMPOSTA UN NUMERO";LIB
   RO(N)
50 NEXTN
60 SCNCLR
70 FORN=0TO13
80 PRINTLIBRO(N)
90 NEXTN
```

Quando darete il RUN al programma, il computer vi richiederà uno alla volta i valori di 14 numeri; poi lo schermo verrà ripulito, e verranno visualizzati tutti e 14 i numeri che avrete introdotto. Eccovi i chiarimenti sul funzionamento di questo programma:

Linea 10: riserva memoria sufficiente ai 14 numeri della variabile multipla (vettore) LIBRO
Linea 20: cancella lo schermo
Linea 30: inizia un ciclo di replicazione delle istruzioni comprese fra FOR e NEXT (14 ripetizioni)
Linea 40: visualizza il messaggio PER FAVORE IMPOSTA UN NUMERO ed attendi l'introduzione di un valore, che sarà assegnato all'elemento N.mo del vettore LIBRO
Linea 50: fissa il termine del ciclo FOR...NEXT
Linea 60: cancella lo schermo
Linea 70: inizia ciclo di replicazioni delle istruzioni comprese fra un nuovo ciclo FOR...NEXT di 14 iterazioni
Linea 80: visualizza il numero contenuto come N.mo elemento della variabile multipla LIBRO
Linea 90: fissa il termine del secondo ciclo FOR...NEXT

Il programma precedente usa una variabile multipla (come si suol dire) uni-dimensionale: continuando il nostro paragone col libro, avevamo a che fare con un singolo libro. Possiamo utilizzare però anche variabili multiple a 2 dimensioni: per continuare il confronto, significa che non solo possiamo specificare la pagina di un libro, ma pure di quale libro fra quelli posti in un certo scaffale. La nostra variabile multipla (o "matrice", come spesso viene chiamata) ha due dimensioni, l'una specifica il libro e l'altra la pagina in questo.

Queste variabili a due dimensioni sono assai utili quando si devono strutturare delle tabelle. Per esempio se volessimo raccogliere i risultati di certe gare di atletica in una tabella, questa potrebbe avere all'incirca la forma:

	Primo	Secondo	Terzo
100 m (maschili)	Paolo	Marco	Franco
400 m (maschili)	Mario	Giorgio	Carlo
100 m (femminili)	Silvia	Giovanna	Clara
400 m (femminili)	Maria	Susanna	Giuliana

Una tabella di questo tipo si può facilmente rappresentare mediante una variabile multipla a due dimensioni (matrice). Ci serve una variabile bidimensionale che possa contenere 12 (= 4 righe di 3 colonne) elementi

(numeri o gruppi di caratteri). In questo caso dovremmo ricorrere ad una matrice di stringhe, perchè vogliamo che contenga dei nomi. Ecco un programma che serve bene allo scopo:

```
 10 SCNCLR
 20 DIMPO$(2,3)
 30 PO$(0,0)="PAOLO":PO$(1,0)="MARCO":PO$(2
    ,0)="FRANCO"
 40 PO$(0,1)="MARIO":PO$(1,1)="GIORGIO":PO$
    (2,1)="CARLO"
 50 PO$(0,2)="SILVIA":PO$(1,2)="GIOVANNA":P
    O$(2,2)="CLARA"
 60 PO$(0,3)="MARIA":PO$(1,3)="SUSANNA":PO$
    (2,3)="GIULIANA"
 70 PRINTTAB(9);"RISULTATI DELLE GARE":PRIN
    T
 80 PRINT"1) 100M (MASCHILI)":PRINT"2) 400M
     (MASCHILI)"
 85 PRINT"3) 100M (FEMMINILI)":PRINT"4) 400
    M (FEMMINILI)"
 90 PRINT:INPUT"DI QUALE GARA VUOI I RISULT
    ATI";GARA
100 IFGARA>4ORGARA<1THENGOTO90
110 INPUT"QUALE POSTO VUOI CONOSCERE";PS
120 IFPS>3ORPS<1THENGOTO110
130 GARA=GARA-1:PS=PS-1
140 PRINT"CHI HA OTTENUTO TALE POSTO NELLA
    GARA   ERA ";
150 PRINTPO$(PS,GARA)
160 PRINT"PREMI UN TASTO PER UN ALTRO RISUL
    TATO"
170 GETKEY A$:SCNCLR:GOTO70
```

Ecco come funziona il programma:

Linea10: cancella lo schermo
Linea20: riserva in memoria un'area sufficiente a contenere una matrice di nome PO$ capace di contenere 3x4 elementi
Linee 30-60: assegna a ciascun elemento della matrice POSTO$ i vari nomi dei gareggianti, p. es. all'elemento POSTO$(0,0) il nome Paolo, all'elemento POSTO$(1,0) il nome Marco, ecc.

Linea 70: visualizza il messaggio RISULTATI DELLE GARE con il primo carattere posto alla 9.a colonna, e quindi esegui un'interlinea
Linea 80: visualizza i testi "100 m (maschili)", "400 m (maschili)", ecc.
Linea 90: interponi una linea vuota, poi visualizza "DI QUALE GARA VUOI I RISULTATI" ed attendi la risposta, che va assegnata alla variabile GARA
Linea 100: se il valore della variabile GARA risulta maggiore di 4, o minore di 1, torna alla linea 90 e ripeti la richiesta
Linea 110: visualizza QUALE POSTO VUOI CONOSCERE ed assegna il risultato alla variabile PS
Linea 120: se il valore della variabile PS risulta maggiore di 3 o minore di 1, torna alla linea 110 e ripeti la richiesta
Linea 130: sottrai 1 dal valore della variabile GARA e riassegna il risultato a GARA, poi sottrai 1 dal valore di PS e riassegna il risultato alla variabile PS
Linea 140: visualizza il testo CHI HA OTTENUTO TALE POSTO NELLA GARA ERA
Linea 150: ricerca il valore dell'elemento (PS, GARA) di POSTO$ e visualizzalo sullo schermo
Linea 160: interponi una riga vuota, poi stampa il messaggio 'PREMI UN TASTO PER UN ALTRO RISULTATO'
Linea 170: attendi che venga premuto un tasto, poi cancella lo schermo e vai alla linea 70 riprendendo da lì il programma

Le dimensioni delle vostre variabili multiple sono limitate solo dall'entita di spazio di memoria disponibile, per cui si possono dimensionare anche "supermatrici" come A(4,4,4,4,4,4) se vi basta lo spazio. Tuttavia, in pratica sarà raro dover utilizzare matrici di più di tre dimensioni, ossia come A(5,5,5).

Sommario

L'istruzione DIM serve a riservare spazio in memoria per una variabile multipla.

Un elemento è un numero, o gruppo di caratteri (stringa) assegnato ad un componente d'una variabile multipla.

Una variabile multipla si definisce tramite un nome seguito (fra parentesi, e separati da virgole) da numeri che rappresentano le diverse dimensioni (massimo valore dell'indice).

Le dimensioni totali di una variabile multipla sono limitate soltanto dalla memoria disponibile.

Come riordinare i programmi

CHR$

Vi sarete probabilmente chiesti in che modo il computer "deposita" i vari caratteri in memoria. Ovviamente, esso non può memorizzare realmente le "forme" dei vari caratteri che appaiono in un programma: in pratica, a ciascuno di essi viene assegnato un codice. Per esempio, la lettera A ha il codice 65 , il simbolo del "cuore" ha il codice 115, e così via.

Il comando CHR$ ci permette di usare i codici dei vari caratteri per visualizzarli. Se battete

PRINT CHR$(65)

vedrete comparire sullo schermo una lettera A, perchè come detto la lettera A ha il codice 65. Analogamente, battendo

PRINT CHR$(115)

vedrete comparire il simbolo del cuore sullo schermo. Potete sperimentare un po' con i vari codici dei caratteri e vederne gli effetti (potete ricavare i numeri di codice fra quelli elencati nell'Appendice C, ma non usate un numero superiore a 255 perchè dopo questo codice non ci sono più caratteri).

Alcuni dei codici usabili con CHR$ sono dei caratteri di controllo, che hanno codici fra 0 e 31, benchè ce ne sia qualcuno anche al di fuori di questi limiti. I caratteri di controllo sono dei caratteri speciali che servono per spostare il cursore, modificare il colore dei testi, e così via.

TAB

Il comando TAB viene utilizzato sempre assieme a PRINT, perchè specifica in quale colonna deve cominciare la visualizzazione o stampa. Per esempio

PRINT TAB(30);"SALVE"

visualizzerà il messaggio SALVE, la cui S comparirà nella 30.a colonna, la A nella 31.a, e così di seguito.

Una cosa importante da ricordare è che se per es. avete battuto l'istruzione

PRINT TAB(30);"SALVE"; TAB(10);"A TUTTI"

dove la seconda parte deve figurare *dopo* la prima, vedrete che sarà così ma la seconda parte comparirà sì a partire dalla colonna 10 come specificato, ma nella riga successiva.

DELETE

Supponete di stare scrivendo un programma abbastanza lungo, e che ad un certo punto decidiate di non aver più bisogno di una certa routine lunga circa 15 linee scritta in precedenza. Per eliminarla dal programma si potrebbe, secondo quanto sappiamo sinora, cancellare una riga per volta battendo il relativo numero di linea e di seguito RETURN, ma la cosa richiede tempo e polpastrelli. Per non consumare eccessivamente questi ultimi, il vostro computer dispone di un comando con il quale potete eliminare tutte in un colpo più linee di programma: il comando DELETE.

Il comando DELETE si usa in modo assai simile a LIST, salvo che invece di visualizzare le varie linee di programma, DELETE le elimina dal programma stesso. Per esempio, se desiderate cancellare le linee da 90 a 150 incluse, dovete battere

DELETE 90—150

Per cancellare le linee dall'inizio del programma sino alla 75 inclusa impostate

DELETE—75

ed infine se volete cancellare dalla linea 5010 in avanti

DELETE 5010—

RENUMBER

Il comando RENUMBER è assai utile nella scrittura dei programmi, perchè consente di "riordinare" i numeri di linea. Supponete per fare un

esempio che abbiate scritto un buon pezzo di programma, numerando secondo la consuetudine le linee di 10 in 10. Successivamente vi accorgete però che fra le linee 50 e 60 vi necessita inserire altre 13 linee di istruzioni. Ovviamente non c'è spazio per 13 numeri, ed allora che fare? Si possono rinumerare, con l'uso del comando RENUMBER, le linee in incrementi di 20, ed ecco che vi rimane spazio per 13 nuove linee fra le (nuove) linee 100 e 120 (anzi, anche per qualcuna di più, se dovesse ancora rendersi necessario in seguito).

Si può usare RENUMBER in vari modi. Ecco i diversi casi:

RENUMBER: rinumera l'intero programma, partendo dalla linea 10 con incremento fisso di 10 per i numeri di linea successivi.

RENUMBER 50,7: rinumera il programma a partire dalla linea 7, che diventa 50, con incremento fisso 10

RENUMBER 100,20,15: rinumera il programma a partire dalla linea 15, che diventa 100, e con incremento fisso di 20

RENUMBER 200,40: rinumera il programma a partire dalla prima linea, che diventa la linea 200, con incremento fisso 40.

REM

Quando si scrive un proprio programma, specie se questo risulta abbastanza lungo, può convenire inserire in certi punti dei commenti che servano a ricordare che cosa fanno le varie parti del programma. Il comando REM permette appunto questo: i commenti inseriti dopo REM in una linea del programma vengono ignorati dal computer. Per fare un esempio, una linea come quella qui sotto potrebbe servire a ricordare che la routine che segue serve per il movimento verso sinistra di una "racchetta":

530 REM SPOSTAMENTO A SINISTRA RACCHETTA

Dopo una REM potete inserire quello che vi pare, tanto il resto della linea verrà ignorato dal computer.

END

Abbiamo già avuto occasione di impiegare END in un paio di casi nei nostri programmi, e dovreste quindi ormai sapere che esso serve a segna-

lare al computer che deve fermare l'esecuzione del programma, anche se non ne ha raggiunto l'ultima linea. Potrebbe sembrare che la cosa non sia utile, ma se alla fine del programma ci sono delle subroutine, dovete poter fermare il programma prima dell'ultima linea.

STOP

Il comando STOP è simile ad END, salvo che con esso è possibile far ripartire il programma dal punto in cui si era fermato. Il computer vi segnala inoltre il numero di linea dove si è fermato.

CONT

È appunto questo comando CONT che serve per far continuare il programma dopo uno STOP, oppure anche dopo che l'avete arrestato in modo diretto premendo il tasto RUN/STOP. Il programma riprenderà l'esecuzione dal punto in cui si era fermato, senza alcuna modifica o reinizializzazione dei valori delle variabili. In certi casi CONT può non funzionare: per esempio, se avete modificato una linea di programma dopo lo STOP. In tali casi sarà visualizzato il messaggio CAN'T CONTINUE ERROR ("non posso proseguire").

Finestre

Le cosiddette "finestre" vi consentono di lavorare entro una certa area dello schermo mentre il resto di esso non viene disturbato. Tutto quello che normalmente può essere effettuato sullo schermo completo viene limitato alla zona della finestra prescelta. Per fare un esempio, cancellate lo schermo (mediante la pressione di SHIFT e CLEAR/HOME), quindi premete per cinque volte il tasto "freccia in basso". Premete ora il tasto ESC e poi la T, quindi premete per 8 volte il tasto "freccia in basso". Premete il tasto ESC e poi la B. Infine premete HOME: il cursore salterà nell'angolo superiore sinistro della finestra che si sarà così definita sullo schermo.

Entro una finestra potete battere un programma, o LISTarlo: ogni cosa che batterete apparirà dentro la finestra, come ben presto vi accorgerete. In pratica, la finestra funziona come una versione in formato ridotto dello schermo intero. Per poter "uscire" dalla finestra dovete prima premere due volte il tasto HOME.

Premendo il tasto ESC seguito da T (Top=cima) si fissa il vertice superiore sinistro della finestra; premendo invece ESC seguito da B (Bottom=-fondo) si definisce il vertice inferiore destro. Noterete come, una volta definito il vertice sinistro, non si può più spostare il cursore in alto o verso sinistra; così come, una volta definita completamente la finestra, il cursore non si può spostare fuori di questa, a meno di non premere prima per due volte di seguito il tasto HOME.

Si possono definire le finestre anche da programma. Occorre conoscere il codice del tasto ESC, che è 27. Per fissare una data finestra da programma si seguirà l'esempio che segue, dove si precisa che [5X CURS GIU] significa che si deve premere a quel punto per 5 volte il tasto "freccia in basso", mentre [4X CURS DES] significa analogamente "premere per 4 volte il tasto freccia a destra".

```
10 SCNCLR
20 PRINT"[5X CURS GIU] [4X CURS DES]";CHR$
   (27);"T";
30 PRINT"[10X CURS GIU] [14X CURS DES]";CH
   R$(27);"B [HOME]";
40 FORN=1TO100:PRINT"FINESTRA!";:NEXTN
```

Come probabilmente noterete, definire una finestra da dentro un programma è molto simile a quando la si definisce con comandi diretti. Le dimensioni e posizioni della finestra sono sempre definite "spostando" il cursore nella posizione ove si vuole il vertice superiore sinistro, e poi nel vertice inferiore destro. L'unica differenza sta nel fatto che i controlli del cursore sono posti fra virgolette; il tasto ESC è sostituito da CHR$(27), e le lettere T e B vanno anch'esse fra virgolette.

Le finestre possono risultare utili per convertire per il vostro computer un programma BASIC scritto per un altro computer. Qualora le dimensioni dello schermo per i testi dell'altro computer non superino le 25 righe per 40 colonne, potete aprire una finestra di dimensioni pari a quello dello schermo dell'altro computer, risparmiandovi buona parte del lavoro di aggiustamento.

Il tasto ESC

Abbiamo appena utilizzato il tasto ESC per definire una finestra sullo schermo, ma questo versatile tasto può venire impiegato in molti altri

modi. Quando il tasto ESC viene usato assieme ad un tasto "letterale", esso serve per attivare alcune speciali funzioni del vostro computer, di cui vi forniamo ora un breve elenco ed una sommaria descrizione di quel che fanno.

ESC A	attiva il modo "inserimento caratteri"
ESC B	fissa il vertice inferiore destro di un finestra nella posizione corrente del cursore
ESC C	disattiva il modo "inserimento caratteri"
ESC D	cancella la linea in cui si trova il cursore
ESC I	inserisce una linea
ESC J	muove il cursore all'inizio della riga corrente
ESC K	muove il cursore alla fine della riga corrente
ESC L	abilita lo "scroll" dello schermo (normalmente ON)
ESC M	disabilita lo "scroll"
ESC N	fa tornare lo schermo alle dimensioni normali
ESC O	disattiva i modi lampeggiamento, caratteri inversi, inserimento e "virgolette"
ESC P	cancella tutto quanto sta sulla riga prima del cursore
ESC Q	cancella tutto quanto sta sulla riga dopo il cursore
ESC R	riduce le dimensioni dello schermo
ESC T	fissa il vertice superiore sinistro di una finestra nella posizione corrente del cursore
ESC V	"Scrolla" in su di una riga
ESC W	"Scrolla" in basso di una riga
ESC X	cancella la funzione ESC

Premendo ESC seguito da A (dovete rilasciare il tasto ESC prima di premere il secondo tasto) si attiva il modo "inserimento". Significa che quanto batterete da quel punto in poi verrà inserito nella riga corrente, a partire dalla posizione del cursore. Se per esempio impostate

10 PRINT "SALVE"

poi spostate il cursore prima del secondo paio di virgolette e battete ESC e poi A, quanto batterete da quel momento verrà inserito prima delle virgolette, le quali si sposteranno verso destra di una posizione ad ogni inserimento di un carattere. Una volta terminato l'inserimento del nuovo testo in una certa linea, premendo ESC e poi C si disattiva il modo "inserimento".

Potete anche cancellare un'intera linea, facendo spostare verso l'alto

("scrollare") di una riga tutto quanto segue per riempire lo spazio resosi vuoto. Per esempio, se date il RUN al programma mono-linea citato sopra e lo lasciate svolgere per un pò, e poi lo fermate, e spostate il cursore su un punto qualsiasi dello schermo e fate ESC seguito da D, vedrete che il contenuto della linea dove stava il cursore sparisce, mentre tutto quanto sta sotto si solleva di una riga.

L'opposto della cancellazione è l'inserimento di una linea. Premendo ESC seguito da I vedrete spostarsi di una riga verso il basso tutto quanto sta sotto la posizione del cursore, lasciando in quel punto una riga vuota per inserire quel che volete.

Certe volte può risultare utile spostare direttamente il cursore all'inizio di una certa linea mentre siete in una posizione intermedia. Il modo più rapido per farlo è premere ESC seguito da J. Per esempio, spostando il cursore in un punto qualsiasi, preferibilmente verso il mezzo d'una riga, e poi premendo ESC e J, il cursore salterà all'inizio di tale riga.

In modo perfettamente analogo si può saltare alla fine della riga dove è il cursore premendo ESC seguito da K.

Può essere assai utile in certi casi sopprimere lo 'scroll' dello schermo. Se premete ESC seguito da M, e spostate il cursore al fondo dello schermo, vedrete che esso ricompare in cima se seguitate a spostarlo verso "il basso". Si tratta della caratteristica che viene pittorescamente chiamata "riavvolgimento" ("wrap-around" in inglese). Se a questo punto LISTate un programma vi accorgerete che quando lo schermo diventa pieno non si ha lo 'scroll' per mostrarvi le righe sucessive, ma invece il listato ricomincia dall'alto dello schermo. Per tornare alla normalità basta premere ESC seguito da L.

Premendo ESC e poi O si ha la disattivazione di qualsiasi lampeggiamento o inversione dei caratteri, nonchè dei modi "inserimento" e "virgolette" che fossero eventualmente in atto. Per esempio, battete CONTROL e FLASH ON seguite da qualche lettera. Ora premete ESC e poi O, battete ancora qualche altra lettera, e vedrete che queste non sono più lampeggianti. Lo stesso accade se fossero attivi i modi "inversione", "inserimento" e "virgolette" (per quest'ultima modalità si intende che dopo aver aperto delle virgolette, premendo un tasto di controllo del cursore compare invece un simbolo).

Quanto viene visualizzato dal computer sul vostro ricevitore televisivo, a volte, può essere di dimensioni troppo grandi per stare entro lo schermo. Il rimedio migliore è cambiare televisore, ma se proprio non ce n'è un

altro potete ridurre le dimensioni dello schermo premendo ESC e poi R : lo schermo si cancellerà, ed il nuovo schermo avrà un bordo ampio un carattere tutto attorno. ESC seguito da N riporta alle condizioni normali.

Se desiderate, si può usare il tasto ESC per scrollare il contenuto dello schermo verso l'alto o verso il basso, di una riga: con ESC più V si ha lo scroll in su, con ESC più W lo scroll in basso.

La funzione ESC dimostra tuttavia forse meglio la sua utilità quando è usata da programma. Avete già visto come ESC possa venire utilizzata per definire una finestra in un programma BASIC, e quindi ormai sapete che per inserirla in un programma dovete scrivere PRINT CHR$(27) e far seguire la lettera tipica del comando diretto fra virgolette, per esempio così:

```
PRINT CHR$(27);"W"
```

che farà scrollare verso il basso di una riga, proprio come se aveste premuto ESC seguito da W.

Queste funzioni speciali possono risultare estremamente utili in un programma, così come in modo diretto. Sarebbe una buona idea imparare ad usarle appropriatamente, con un pò di prove: sarete sorpresi delle volte in cui tornano utili.

Tutti facciamo degli errori

"Trappole" per gli errori

Perchè un programma si possa dire "buono" occorre che esso sia "amichevole" ("user-friendly", dicono gli anglosassoni). Il termine significa che chi utilizza il programma riceve ampie istruzioni da parte del computer per un uso corretto del programma, e che qualsiasi cosa, per quanto... stupida, che l'utente può fare non blocca il programma, ma questo invece informa l'utente sull'errore commesso. Per aiutarvi a scrivere così i vostri programmi, il vostro computer è stato dotato di apposite "trappole" contro gli errori, per cui se ad es. premete in modo inopportuno il tasto RUN/STOP, il programma vi spiegherà dove avete sbagliato.

Eccovi un breve programma esemplificativo:

```
 10 TRAP90
 20 SCNCLR
 30 PRINT"NON PREMETE PER FAVORE IL TASTO RU
    N/STOP"
 40 PRINT"MENTRE STAMPO LE 'O' SULLO SCHERM
    O."
 50 FORN=1TO500:NEXTN
 60 FORN=1TO1024:PRINT"O";:NEXTN
 70 FORN=1TO500:NEXTN
 80 RUN
 90 SCNCLR
100 IFER=30THENPRINT"TI AVEVO PREGATO DI NO
    N PREMERE QUEL     TASTO!":ELSESTOP
110 PRINT"ADESSO RIPRENDERO L'ESECUZIONE, M
    A"
120 PRINT"PERFAVORE, NON PREMERE RUN/STOP."
130 FORN=1TO2000:NEXTN
140 RESUME20
```

Nella linea 10 di questo programma troviamo il primo esempio di comando per "intrappolare" gli errori: TRAP. Questo comando dice al computer a quale linea deve saltare nel caso insorga una situazione di errore: nel nostro caso, di saltare alla linea 90, e di lì proseguire l'esecuzione.

Il resto del programma è abbastanza semplice, almeno sino alla linea 100. In questa troviamo il riferimento ad una variabile ER, che non è stata assegnata prima. Si tratta in realtà di una "variabile di sistema", ossia di una particolare variabile utilizzata dal sistema operativo del computer. ER contiene il numero dell'ultimo errore manifestatosi. Dato che il numero di errore per aver premuto il tasto RUN/STOP (per il computer, anche questo è un particolare caso di "errore") è 30, il computer passerà ad eseguire dalla linea 100 il programma solo se è stato premuto RUN/STOP.

Anche nella linea 140 troviamo un nuovo comando. Il comando RESUME permette al computer di continuare l'esecuzione del programma principale dopo che è insorto un errore. RESUME assomiglia a GOTO, con la differenza che esso fissa il termine della routine di "intrappolamento" degli errori: per cui quando il computer giunge alla linea 140 si accorge che si tratta della fine della routine di trattamento degli errori, e salta così alla linea 20 per continuare da lì l'esecuzione del programma.

Esiste un'altra forma del comando RESUME, ed è RESUME NEXT. Con questa istruzione si dice al computer di passare al programma principale e di riprenderlo a partire dall'istruzione immediatamente successiva a quella ove si era manifestato l'errore. Per esempio, modificata la linea 140 in

```
140 RESUME NEXT
```

e dato poi RUN, premete ad un certo punto il tasto RUN/STOP. Vedrete comparire il solito messaggio che non avreste dovuto premere quel tasto, e, dopo una breve pausa, il computer continuerà il programma dal punto in cui si era fermato.

C'è anche un'altra variabile di sistema che viene impiegata per gli errori. Si tratta della variabile EL, che contiene il numero di linea dove è occorso l'ultimo errore. Così, se cambiate la linea 10 in

```
10SALVE
```

e date il RUN, avrete subito la comparsa di un errore (ovvio, dato che la sintassi è sbagliata). Se ora impostate

```
PRINT EL
```

il computer visualizzerà il numero 10, ossia quello della linea dove l'errore si è manifestato.

Sappiamo che la variabile di sistema ER contiene invece il numero (=codice) di errore , che però non è molto indicativo della natura dell'errore stesso. Per esempio, se vi viene segnalato che è stato commesso l'errore numero 11, non ne saprete molto di più...Fortunatamente, esiste una funzione che può esserci d'aiuto in casi come questi: ERR$. Se battete

PRINT ERR$(11)

il computer segnalerà sullo schermo SYNTAX (errore di sintassi): infatti l'11 corrispnde a SYNTAX ERROR. Similmente, se battete

PRINT ERR$(14)

avrete in risposta il messaggio ILLEGAL QUANTITY. Potete impiegare ERR$ per individuare la natura dei vari tipi di errore, eccetto quelli relativi all'unità disco.

Sommario

Risulta possibile (con TRAP) "intrappolare" un errore: quando dovesse occorrerne uno, il computer salterà alla vostra routine di trattamento dell'errore.

Il comando RESUME indica dove finisce la vostra routine di trattamento degli errori, e dice al computer a quale linea deve saltare per riprendere l'esecuzione del programma.

La variabile di sistema ER contiene il numero del tipo di errore intervenuto per ultimo.

La variabile di sistema EL contiene invece il numero di linea dell'istruzione per la quale l'ultimo errore si è manifestato.

Per trovare di quale errore si tratta in modo più esplicito, si può visualizzare il relativo messaggio con la funzione ERR$.

HELP

Il comando HELP risulta assai utile quando state cercando di rintracciare uno sbaglio in una linea di programma. Se per esempio avete una linea che contiene 4 o 5 comandi diversi, e sapete che in essa si è manifestato un errore ma non sapete dove, dovete semplicemente battere HELP e vedrete comparire la linea in questione sullo schermo Il comando responsabile

dell'errore verrà segnalato lampeggiando, per cui risulta facilissimo da individuare. Il comando HELP funziona soltanto dopo che è stata emesso un messaggio di errore. Premere il tasto HELP equivale a battere HELP per esteso.

TRON e TROFF

È assai raro che un programma funzioni bene al primo colpo, quale ne sia la lunghezza. Rintracciare i veri errori (quelli che vengono identificati e segnalati dal computer) non è difficile, specie con l'ausilio del comando HELP. Però spesso in un programma si verificano degli errori più occulti che, benchè il programma "giri" regolarmente, fanno sì che il programma non faccia esattamente quello che ci si attende da esso.

Per poter eliminare anche questi "bug" (letteralmente : bachi), come vengono spesso chiamati in informatica (per cui le operazioni di ricerca ed eliminazione di tali fonti di errore passano in gergo sotto il nome di "debugging"), il computer è stato dotato di due altri utili comandi: TRON e TROFF. Il primo (da TRace ON) attiva la funzione di "tracciamento": il che significa che il computer durante l'esecuzione visualizza in successione sullo schermo il numero della linea che sta correntemente eseguendo, per cui quando interviene un eventuale errore ci si può immediatamente rendere conto del punto in cui il fatto è accaduto.

Per esempio, se vogliamo che il ... messaggio prodotto dalla linea 20 del seguente programma sia modificato in SALCICCIA E MOSTARDA invece di TONNO E FAGIOLI potremmo usare allo scopo il comando TRON per ricercare la linea in corrispondenza alla quale il messaggio viene emesso (so bene che in un programma così corto la linea si rintraccia a prima lettura, ma rendetevi conto che ci possono essere in pratica programmi lunghi qualche centinaio di linee dove la ricerca a vista risulta assai più difficile: questo è solo un esempio). Quindi, impostate il programmino, poi battete TRON e date RUN:

```
10SCNCLR
20PRINT"TONNO E FAGIOLI"
30FOR I = 1 TO 10
40PRINT"QUESTA È UNA PROVA";
50NEXT I
```

Mentre ciascuna linea viene eseguita, vedrete comparire sullo schermo il numero di linea fra parentesi quadre ([ ]). Quando compare [20] vedrete anche il messaggio TONNO E FAGIOLI, così saprete imediatamente che la linea che produce il messagio è la numero 20. Il "tracciamento" rimane in funzione sino a quando non lo disattivate con TROFF (TRace OFF).

Programmazione avanzata

READ, DATA e RESTORE

Spesso risulta utile poter disporre di una lista di numeri o caratteri a cui il computer può fare riferimento ed utilizzare. Essi possono costituire una lista di nomi ed indirizzi, in cui per esempio volete far ricercare al computer un dato nominativo. Una delle maniere per ottenerlo è di memorizzare ciascun nome ed indirizzo di ogni persona in una variabile stringa, o magari in una variabile stringa multipla (vettore di stringhe), e cercare fra queste variabili quando si vuol trovare un dato nominativo. Un modo più facile e comodo, però, è offerto da una lista di DATA, nelle quali il computer può cercare il nome desiderato. Per definire una simile lista si usa il comando DATA, nel modo qui esemplificato:

1000 DATA "MARIO ROSSI", 123456, "CARLO BIANCHI", 123642

Si tratta in questo caso di una lista assai corta, che contiene solo due nomi e relativi numeri di telefono, ma la lista potrebbe continuare, usando se del caso più di una linea di DATA. Come potete notare da questo esempio, nella lista si può includere qualsiasi tipo di carattere, e i caratteri possono essere posti fra virgolette, anche se questo non è essenziale. Nella lista si possono inserire anche numeri, e si possono mescolare a piacimento numeri e caratteri, come potete vedere.

Naturalmente, una lista così serve a poco da sola, se non se ne può fare qualcosa. Quel che possiamo fare è "leggere" (READ) la lista, assegnando ogni singolo dato ad una variabile: i numeri ad una variabile numerica, e gli altri caratteri ad una variabile stringa (se lo si vuole, anche i numeri possono essere assegnati ad una variabile stringa, ma naturalmente non si possono allora fare con essi dei calcoli, almeno non così semplicemente). Il programma che segue serve per un semplice repertorio telefonico, e impostando il nome di una persona il computer fornisce il suo numero di telefono.

```
10 SCNCLR:INPUT"COME SI CHIAMA LA PERSONA"
   ;NOME$
20 FORN=1TO5:READA$,X
30 IFA$=NOME$THENPRINT"IL NUMERO DI TELEFO
   NO E'";X:GOTO60
40 NEXTN
50 PRINT"SPIACENTE, NON CONOSCO IL NUMERO
   DI      QUESTA PERSONA."
60 PRINT:PRINT"PREMI UN TASTO"
70 GETKEY A$:RUN
80 DATAMARIO ROSSI,123456,CARLO BIANCHI,23
   4654,GIORGIO VERDI,954186
90 DATAFRANCO NERI,437293,MICHELE AZZURRI,
   284916
```

Se date il RUN al programma lo schermo si cancellerà, e quindi vi verrà chiesto di impostare il nome della persona di cui volete conoscere il numero di telefono (scegliete un nome fra quelli compresi nella lista dei DATA). Il computer scorrerà allora attraverso la lista di DATA e se trova una persona il cui nome coincide con quello richiesto vi segnalerà il suo numero di telefono (che nella lista è indicato immediatamente dopo il nome). Se il computer non trova il nominativo indicato, ve lo segnalerà.

Ecco come funziona il programma:

Linea 10: cancella lo schermo; visualizza il messaggio "COME SI CHIAMA LA PERSONA" ed attendi la risposta, che verrà assegnata alla variabile stringa NOME$
Linea 20: inizia la replicazione delle istruzioni comprese fra FOR e NEXT (per 5 volte), col valore di N che parte da 1 e viene incrementato di 1 ad ogni passaggio, sino a quando raggiunge 5. Leggi (READ) il successivo dato dalla lista di DATA, ed assegnalo alla variabile stringa A$, quindi leggi anche il dato seguente ed assegnalo alla variabile X.
Linea 30: verifica se la variabile stringa A$ coincide con quella NOME$: se sì, visualizza il messaggio "IL NUMERO DI TELEFONO È" e quindi visualizza il valore della variabile X, prima di saltare alla linea 60 per proseguire da lì l'esecuzione.
Linea 40: segna la fine del ciclo FOR...NEXT
Linea 50: visualizza il messaggio "SPIACENTE, NON CONOSCO IL NUMERO DI QUESTA PERSONA"
Linea 60: inserisci una riga vuota (interlinea) e poi visualizza "PREMI UN TASTO"

Linea 70: attendi la pressione di un tasto, ed assegna il simbolo posto su tale tasto alla variabile A$ prima di ricominciare il programma
Linea 80: lista di DATA **Linea 90**: seguito lista di DATA.

La linea 20 procede a leggere (READ) il dato successivo dalla lista di DATA posta alla fine del programma (in realtà, vengono letti due dati successivi, ma uno alla volta). Ogni volta che vien letto un dato, il computer "ricorda" a che punto della lista è arrivato, per cui al prossimo comando READ provvede a leggere il dato successivo.

Con una singola READ si possono leggere più di un dato, come si può vedere anche dal programma appena illustrato. Tutto quel che si deve fare è di specificare al computer le diverse variabili a cui ciascun dato, letto nell'ordine corretto, va assegnato, e separare ciascuna variabile della lista con una virgola. Potete leggere quanti dati volete, almeno sino a che la corrispondente lista di variabili di assegnazione sta in una normale linea di programma: se questa non bastasse, si useranno due linee con due distinti READ.

Ma cosa accade quando il computer raggiunge la fine dei DATA? Se il computer giunge in fondo alla lista e non trova più DATA per un READ, emette il messaggio di errore OUT OF DATA ERROR, se cercate di leggere (READ) altri dati. Se per esempio modificate la linea 70 del programma precedente in

```
70GETKEY A$:GOTO 10
```

e date il RUN, ad un certo punto vedrete comparire il messaggio OUT OF DATA ERROR. Ciò dipende dal fatto che il RUN reinizializza pure il puntatore dei DATA (su cui si basa il computer per ricordare dove ed in che punto della linea di DATA sta) all'inizio della lista, mentre un GOTO non lo fa. Per ovviare a questo problema si usa il comando RESTORE, che dice al computer di iniziare da quel momento la lettura dei DATA dal principio della lista. Se cambiate la linea 70 così:

```
70 GETKEY A$:RESTORE:GOTO 10
```

il programma funzionerà regolarmente.

Potete anche ordinare al computer di iniziare il READ a partire da una certa linea di DATA. Per esempio, provate a modificare ancora la linea 70 in

```
70GETKEY A$:RESTORE 90:GOTO 10
```

Il RESTORE 90 dice al computer di cominciare la lettura (=fissare inizialmente il puntatore dei DATA) alla linea 90. Ora vi accorgerete che siete in grado di trovare soltanto gli indirizzi di Franco Neri e Michele Azzurri.

Vi sarete probabilmente immaginati a questo punto che quando il computer trova un'istruzione READ salta alla linea che contiene i DATA e ne esamina i dati uno alla volta. Non è proprio così: per convincersene, battete il comando TRON e poi date il RUN al programma. Vedrete che il computer non effettua mai una salto alle linee 80 e 90 dove sono i DATA. Il computer in effetti sa benissimo dove trovare in memoria i dati successivi, e non ha bisogno di preoccuparsi dei numeri di linea, ma va a ricercarli direttamente in memoria.

Sommario

L'istruzione DATA viene usata per contenere una lista di carararatteri e/o numeri. I caratteri possono essere racchiusi fra virgolette, ma ciò non è essenziale.

L'istruzione READ serve per leggere un singolo valore fra i DATA, assegnandolo ad una variabile. Un singolo READ può però leggere contemporaneamente più dati, assegnandoli a distinte variabili. In tal caso le variabili nell'istruzione READ vanno separate da virgole.

L'istruzione RESTORE dice al computer di ripartire nella lettura dei DATA dal primo di essi.

Il computer in pratica non salta alla linea che contiene i DATA quando deve eseguire un READ: esso sa esattamente dove sono memorizzati i DATA e non ha bisogno dei numeri di linea di tali istruzioni.

Manipolazione delle stringhe

Le variabili stringa sono molto utili, anche perchè esistono vari modi in cui potete suddividerle e riarrangiarle a seconda di quel che vi serve. I comandi per manipolare in questo modo le variabili stringa sono ilustrati nelle pagine che seguono.

LEFT$ Se vi ricordate quanto detto nel capitolo su IF...THEN...ELSE, avrete presenti tre o quattro linee simili a questa:

130 IF A$="SI" OR A$="NO" THEN...

Sarebbe più comodo se potessimo limitare la nostra verifica alla prima lettera di A$ per vedere se è una S, perchè allora potremmo accettare indifferentemente come "affermative" risposte del tipo SI,S,STA BENE, SICURO od ogni altra versione per SI, purchè inizi per S. Possiamo usare a questo scopo LEFT$, come in questo esempio:

130IF LEFT$(A$,1)="S" THEN...

Questa particolare linea verifica se il primo carattere di A$ una S. Se la modificate in

130IF LEFT$(A$,2)="SI" THEN...

la verifica verrebbe estesa ai primi due caratteri della variabile A$ (come specificato dal 2 entro le parentesi: se questo fosse un 3 si tratterebbe dei primi 3 caratteri, e così via).

Naturalmente, non è necessario che LEFT$ (o qualsiasi altra delle funzioni stringa che vedremo oltre) sia applicato solo alle variabili stringa: essa si può utilizzare anche con delle stringhe direttamente composte da caratteri posti fra virgolette, come in

250IF LEFT$("COMPUTER",4)="COMP" THEN...

Eccovi un piccolo programma esemplificativo:

```
10 SCNCLR:INPUT"TI PIACE USARE I COMPUTERS
   ";CO$
20 IFLEFT$(CO$,1)="S"THENPRINT"QUESTO MI F
   A MOLTO PIACERE!"
30 IFLEFT$(CO$,1)="N"THENPRINT"OH, MI HAI
   COLPITO NELL'ORGOGLIO!"
```

RIGHT$

Il funzionamento di RIGHT$ è assai simile a quello di LEFT$, con la differenza che quest'ultimo esamina i primi caratteri di una stringa, mentre RIGHT$ verifica quali sono gli ultimi caratteri della stringa. Provate con questo esempio:

```
10 SCNCLR:INPUT"BATTI QUALCOSA, PERFAVORE"
   ;ZZ$
20 PRINT"I PRIMI 2 CARATTERI CHE HAI BATTU
   TO      SONO ";
30 PRINTLEFT$(ZZ$,2):PRINT"GLI ULTIMI DUE
   CARATTERI CHE HAI BATTUTOSONO ";
40 PRINTRIGHT$(ZZ$,2)
```

Il 2 inserito fra le parentesi dopo RIGHT$ dice al computer che si vogliono considerare gli ultimi 2 caratteri della stringa, e può naturalmente assumere anche altri valori secondo necessità, proprio come abbiamo visto per LEFT$.

MID$

MID$ si utilizza per esaminare i caratteri intermedi di una stringa, invece che i primi od ultimi caratteri. Invece di limitarsi ad indicare il numero dei caratteri, come in LEFT$ e RIGHT$, qui occorre anche specificare da che posizione bisogna cominciare. Per esempio, con una linea come questa

310A$=MID$(B$,4,3)

il computer assegnerà alla variabile A$ 3 caratteri interni della variabile stringa B$, partendo dal 4.o carattere di B$ (ossia la stringa composta dal 4°, 5° e 6° carattere).

Tramite MID$ è anche possibile sostituire parte di una variabile stringa . Eccovi un semplice esempio di questa tecnica:

```
10 SCNCLR
20 A$="SALVE SALVE CARISSIMI!"
30 PRINTA$
40 MID$(A$,7,5)="AMICI"
50 PRINTA$
```

Se esaminate la linea 40 vedrete come viene impiegato MID$ per sostituire il secondo SALVE con AMICI. Lo si ottiene semplicemente dicendo al computer da che posizione della variabile stringa si deve cominciare (qui: il settimo carattere) e quanti caratteri vanno sostituiti (nel nostro caso: cinque caratteri), ed infine quali caratteri vanno sostiuiti ai precedenti. Come potete notare, i "nuovi" caratteri vanno posti fra parentesi.

INSTR

L'istruzione INSTR serve per trovare se una certa stringa è contenuta IN un'altra STRinga. Provate con questo programmino:

```
10SCNCLR:A$="PAOLO PAPERINO PARTE PRESTO
PER PARIGI"
20PRINT INSTR(A$,"PAR")
```

Se date il RUN al programma, vedrete comparire sullo schermo il numero 16; infatti la P di PAR compare (per la prima volta) nella 16.a posizione (inizio di PARTE) nella stringa A$. Si è ordinato al computer di esaminare la stringa, ricercando se vi erano contenute in successione le lettere PAR. In caso affermativo, il computer vi segnala l'esatta posizione dove tale gruppo di lettere compare in A$.

Se osservate la stringa A$, noterete che il gruppo di lettere PAR compare due volte, una in PARTE e l'altra in PARIGI: il computer segnala però sempre la prima occorrenza soltanto della stringa ricercata. Se volete identificare anche il secondo punto dove sta PAR, dovete scrivere

```
20PRINT INSTR(A$,"PAR",18)
```

In questo caso il computer ricercherà il gruppo di caratteri PAR in A$ a partire dalla 18.a posizione: dando il RUN vedrete così comparire sullo schermo il numero 33. In altre parole, il computer questa volta ha esaminato la stringa RTE PRESTO PER PARIGI alla ricerca della prima comparsa del gruppo PAR, e l'ha trovato in corrispondenza al 33.carattere, la P di PARIGI, della variabile stringa A$.

Se il computer non rintraccia il gruppo di caratteri che gli è stato specificato, il valore ritornato è il numero 0.

LEN

L'istruzione (funzione) LEN serve a determinare la lunghezza di una variabile stringa, ossia il numero di caratteri di cui è composta (inclusi eventuali spazi). Eccovi un esempio di applicazione:

```
10SCNCLR:A$="PRECIPITEVOLISSIMEVOLMENTE"
20PRINT"LA VARIABILE STRINGA ";A$;" CONTIENE ";
30PRINT LEN(A$);"CARATTERI"
```

Come potete osservare, anche con LEN la variabile stringa di cui si vuole la lunghezza va posta fra parentesi.

Sommario

La funzione LEFT$ si usa per ottenere i primi caratteri di una (variabile) stringa.

Analogamente la funzione RIGHT$ si usa per ottenere gli ultimi caratteri di una (variabile) stringa.

MID$ serve invece per ricavare i caratteri intermedi di una (variabile) stringa.

La funzione INSTR serve ad indicare la posizione in cui, in una data stringa, è contenuta un'altra stringa: il numero fornito in risposta indica la posizione occupata dal primo carattere della stringa ricercata. Se la stringa cercata non è contenuta nella stringa esaminata, il risultato è 0.

La funzione LEN serve a determinare la lunghezza (in caratteri) di una (variabile) stringa, che va posta fra parentesi dopo LEN.

SOUND e VOL

È ormai tempo di cominciare ad esplorare le capacità sonore del vostro computer. Esso è dotato di tre "voci" (generatori sonori), e ne potete impiegare due assieme (vedi oltre). Così una voce può suonare la melodia, e un'altra il ritmo. Due dellle tre voci generano note sonore, la terza produce il cosiddetto "rumore bianco" (utile per simulare spari ed esplosioni).

Prima però di cominciare a generare qualsiasi suono, dobbiamo sapere come regolarne il volume sonoro. Per questo si usa il comando VOL, seguito da un numero che può andare da 0 ad 8 (8 significa livello massimo, 0 minimo, ovvero inaudibile). In genere conviene fissare sempre il volume al massimo, per cui battete

```
VOL 8
```

Ora dovete scegliere una voce ed una nota, e decidere quanto deve durare l'emissione della nota. Se per esempio volete generare la nota C (o DO nella nostra notazione musicale) dalla voce n.o 1 e mantenerla per 3 secondi, scriverete il comando

```
SOUND 1, 810, 180
```

Il numero 1 specifica la "voce" prescelta; il numero 810 indica il DO della terza ottava, ed il terzo valore, 180, indica la durata di 3 secondi (la durata è espressa in sessantesimi di secondo).

Provate a modificare l'1 in 2 e poi in 3, per sentire come differiscono le diverse voci.

Come detto prima, si possono suonare contemporaneamente due voci. Potete usare assieme le voci 1 e 2, oppure 1 e 3. Se battete la linea che segue sentirete suonare una nota su uno sfondo di rumore bianco:

SOUND 1,810,360:SOUND 3,917,360

Il comando SOUND si può impiegare per suonare delle ariette, e per creare degli effetti sonori. Eccovi alcuni programmi che ve ne danno un esempio.

Motivo "Doctor Foster"

```
 10 SCNCLR:VOL8:PRINT"DOCTOR FOSTER"
 20 FORN=1TO36:READNO,LU
 30 SOUND1,NO,LU
 40 NEXTN
 50 FORN=1TO5000:NEXTN
 60 RUN
 70 DATA739,60,739,30,834,60,834,30,810,30,
    834,30,810,30,798,60,770,30
 80 DATA739,60,739,30,798,30,770,30,739,30,
    770,90,770,60,770,30,739,30
 90 DATA739,30,739,30,834,30,810,30,798,30,
    810,30,834,30,810,30,854,30
100 DATA834,30,810,30,798,30,798,30,798,30,
    880,60,770,30,881,90,881,60
```

Telefono

```
10 SCNCLR
20 VOL8
30 FORM=1TO10
40 FORN=1TO10:SOUND1,650,1:SOUND1,700,1:NE
   XTN
50 FORN=1TO50:NEXTN:FORN=1TO10:SOUND1,650,
   1:SOUND1,700,1:NEXTN
60 FORN=1TO1000:NEXTN,M
70 FORN=1TO8:SOUND1,650,1:SOUND1,700,1:NEX
   TN
80 SOUND1,920,20
90 FORN=8TO0STEP-1:VOLN:FORM=1TO5:NEXTM,N
```

Mentre emette un suono, il computer può seguitare a svolgere istruzioni,

per cui si può avere un'aria che suona in sottofondo mentre avviene qualche altra azione.

Sommario

Il livello di volume sonoro si seleziona con il comando VOL, seguito da un valore da 0 ad 8.

Sono disponibili tre "voci" sonore, e si possono usare simultaneamente le voci 1 e 2, oppure 1 e 3.

La durata di una nota può assumere un qualsiasi valore fra 0 e 65535, in sessantesimi di secondo.

Esistono 1024 suoni (note) diversi che posono essere suonati. Essi sono in parte indicati dalla tabella che segue:

NOTA	FREQUENZA (Hz)	VALORE
A (=LA)	110	11
B (=SI)	123.5	122
C (=DO)	130.8	173
D (=RE)	146.8	266
E (=MI)	164.7	349
F (=FA)	174.5	387
G (=SOL)	195.9	457
A (=LA)	220.2	520
B (=SI)	246.9	575
C (=DO)	261.4	600
D (=RE)	293.6	647
E (=MI)	330	689
F (=FA)	349.6	708
G (=SOL)	392.5	743
A (=LA)	440.5	774
B (=SI)	494.9	802
C (=DO)	522.7	814
D (=RE)	588.7	838
E (=MI)	658	858
F (=FA)	699	868
G (=SOL)	782.2	885
A (=LA)	880.7	901
B (=SI)	989.9	915
C (=DO)	1045	921

D (=RE)	1177	933
E (=MI)	1316	943
F (=FA)	1398	948
G (=SOL)	1575	957

Questa tabella copre quattro ottave, ma senza i diesis ed i bemolle. La frequenza di ciascuna nota è indicata solo a scopo di riferimento. I "valori" della terza colonna sono quelli che dovete usare come terzo parametro dopo SOUND: così p. es. per suonare la nota C = DO (terza dall'alto nella tabella) per la durata di mezzo secondo si batterà

SOUND 1,173,60

È possibile suonare una nota di frequenza qualsiasi, o quasi. Se conoscete la frequenza, il valore da usare come parametro di SOUND è dato dalla formula

VALORE = 1024 -(110840.45/frequenza)

ON...GOTO e ON...GOSUB

Nel capitolo dove abbiamo descritto le istruzioni GOTO e GOSUB vi è stato detto che con questi comandi non si può usare una variabile al posto del vero numero di linea. Per ovviare a questo piccolo intralcio, il vostro computer dispone delle istruzioni ON...GOTO e ON...GOSUB. Con essi si avrà il salto ad certo numero di linea o ad una certa subroutine in funzione del valore assunto da una certa variabile. Per esempio, se il computer incontra una linea del tipo

120 ON ZZ GOTO 1000, 2000, 3000, 4000

esso controllerà qual'è il valore correntemente assegnato alla variabile ZZ e quindi salterà ad una delle linee il cui numero è fra quelli che seguono questo particolare GOTO. Precisamente, se il valore di ZZ è 1, esso salterà alla linea 1000; se è 2 salterà alla linea 2000, e così via: in ogni caso alla linea il cui numero occupa nella lista la posizione rappresentata dal valore della variabile.

Il comando ON...GOSUB opera esattamente allo stesso modo, ovviamente saltando ad una fra varie subroutine il cui numero di linea iniziale compare nella "lista" che segue al GOSUB. Ciascuna subroutine deve naturalmente terminare col regolare RETURN, come nel caso del GOSUB normale.

Eccovi un programma che simula il lancio di un dado ed usa, a scopo illustrativo, l'istruzione ON...GOSUB

```
 10 SCNCLR
 20 FORN=9TO13:CHAR1,17,N,"[CONTROL5 CONTRO
    L9]       ":NEXT
 30 DADO=INT(RND(0)*6)+1:IFDADO=7THENGOTO30
 40 ONDADOGOSUB60,70,90,110,130,150
 50 FORN=1TO1000:NEXT:RUN
 60 CHAR1,19,11,"*":CHAR1,17,14,"UNO":RETUR
    N
 70 CHAR1,18,10,"*":CHAR1,20,12,"*"
 80 CHAR1,17,14,"DUE":RETURN
 90 CHAR1,18,10,"*":CHAR1,19,11,"*":CHAR1,2
    0,12,"*"
100 CHAR1,17,14,"TRE":RETURN
110 CHAR1,18,10,"**":CHAR1,18,12,"**"
120 CHAR1,17,14,"QUATTRO":RETURN
130 CHAR1,18,10,"**":CHAR1,19,11,"*":CHAR1,
    18,12,"**"
140 CHAR1,17,14,"CINQUE":RETURN
150 CHAR1,18,10,"**":CHAR1,18,11,"**":CHAR1
    ,18,12,"**"
160 CHAR1,17,14,"SEI":RETURN
```

Là dove nella linea 20 trovate le parentesi quadre, significa che dovete a quel punto, premendo CONTROL 5 e poi CONTROL 9, modificare il colore del testo in porpora, e selezionare i caratteri invertiti.

Eco come funziona questo programma:

Linea 10: cancella lo schermo
Linea 20: modifica il colore del testo in porpora, e disegna un blocco compatto largo 5 caratteri ed alto 5 righe, il cui primo quadratino si trova alla riga N e colonna 17
Linea 30: scegli un numero a caso fra 0 ed 1, moltiplicalo per 6 ed arrotondalo all'intero prima di sommare 1. Il valore finale lo assegni alla variabile DADO. Se tale valore risulta 7 torna ad eseguire questa linea (in definitiva, scegli un numero a caso fra 1 e 6: il caso accennato in cui il risultato iniziale, a cui si somma 1, vale 6, accade raramente)

Linea 40: se il valore di DADO è 1 vai (GOSUB) alla routine che inizia dalla linea 60. Se il valore di DADO è 2 GOSUB 70; se è 3 GOSUB 90, e così via
Linea 50: un ciclo FOR...NEXT di attesa
Linea 60: visualizza un singolo pallino nel quadretto alla riga 11 e colonna 19, poi il testo UNO con la U alla riga 14 e colonna 17, prima di tornare all'istruzione immediatamente successiva al GOSUB che aveva provocato il salto a questa subroutine
Linea 70: visualizza un pallino alla riga 10 e colonna 18, ed un altro alla riga 12 e colonna 20
Linea 80: visualizza DUE con la D posta alla riga 14 e colonna 17

Tutte le linee succesive funzionano in modo analogo alle ultime descritte: esse servono cioè a visualizzare il punto e il relativo valore con una parola.

AUTO

Se state impostando un listato di programma che avete ricavato da una rivista od un libro, può risultare ben noioso dover battere continuamente i vari numeri di linea, specie se questi si succedono regolarmente con incremento fisso (come 10, 20, 30, ecc.). In questo e simili casi, per risparmiarvi un pò di fatica della battitura, il computer vi viene in aiuto con il comando AUTO, ovvero con la numerazione automatica delle linee. Se battete

AUTO 10 e provate a battere un programma, vedrete che ogni volta che premete il tasto di RETURN alla fine di ogni linea, compare da solo il prossimo numero di linea.

Il numero posto dopo AUTO indica al computer l'incremento da dare ai successivi numeri di linea: se fate p. es. AUTO 50, i numeri si succederanno come 50,100,150, e così via.

Una volta che avete terminato il programma, per uscire dalla numerazione automatica (ed anche nel caso lo vogliate nel corso dell'impostazione, per qualche motivo) basta che battiate solo RETURN. Così, se il computer dopo impostata la linea 1510 che conclude il programma vi presenta il prossimo numero 1520, battete semplicemete RETURN e nient'altro.

CLR

In certi casi può rendersi necessario "azzerare" i valori di tutte le variabili nel mezzo dell'esecuzione di un programma. Il modo più comodo è di

usare il comando CLR: il programma non viene nè fermato nè modificato in alcun modo.

Il comando CLR viene eseguito automaticamente quando si modifica una linea di programma, oppure quando si dà il RUN.

ASC

Spesso può tornare utile trovare il codice CHR$ di un certo carattere. Fortunatamente il computer dispone del comando ASC che gli consente di determinare quale è il codice di qualsiasi carattere, senza dover consultare una apposita tabella.

Se impostate

PRINT ASC("A")

vedrete comparire il numero 65 sullo schermo. Come sappiamo, 65 è il codice ASCII della lettera A. Allo stesso modo potete determinare il codice di qualsiasi carattere o simbolo, che dovrà esere incluso fra virgolette e fra parentesi, come nell'esempio citato.

VAL

Il comando VAL è in effetti una funzione, che ritorna il valore numerico di una (variabile) stringa. Per esempio, se alla variabile BS$ è stato in precedenza assegnato il gruppo di caratteri "921", l'istruzione

Z = VAL(BS$)

assegna alla variabile numerica Z il valore 921.

Se nella variabile stringa esiste una combinazione di numeri e lettere, i casi sono due. Se la stringa comincia con un numero, il valore ritornato da VAL è quel numero (trascurando i caratteri non numerici seguenti). Se invece comincia con un carattere non numerico, il valore ritornato è zero: così PRINT VAL("A12") fornisce 0.

STR$

STR$ è l'opposto di VAL, vale a dire che tale funzione converte un

numero in una stringa di caratteri corrispondente. Per esempio, una linea come la

```
320 A$=STR$(864)
```

assegna alla variabile stringa A$ i caratteri '864'.

Il comando STR$ ha la caratteristica di aggiungere sempre uno spazio in testa al numero nella stringa di caratteri a cui viene assegnato. Per esempio, se impostate il programma

```
10SCNCLR:A$=STR$(864)
20PRINT A$:PRINT LEN(A$)
30A$=RIGHT$(A$,3)
40PRINT A$:PRINT LEN(A$)
```

e lo fate girare, vedrete che quando il numero è stato assegnato la prima volta ad A$ gli è stato aggiunto davanti uno spazio. Nella linea 30 tale spazio viene eliminato, e ne avrete la conferma quando il numero verrà nuovamente visualizzato accanto al numero di caratteri che ora compongono la stringa a$.

Variazioni di stampa e la grafica

PRINT USING

Si tratta di una variante del comando PRINT che risulta particolarmente utile quando si vogliono realizzare delle presentazioni di tabelle o diagrammi.

Supponiamo per fare un esempio che in una certa colonna di una tabella si vogliano visualizzare un massimo di 6 cifre. Per ottenere questo, si userà una linea del tipo

760 PRINT USING"######"; NU

Questa istruzione provvede ad arrotondare all'intero il valore di NU (se questo aveva anche una parte decimale), e ad aggiungervi se necessario degli spazi davanti sino a raggiungere il numero totale di 6 caratteri. Così, se NU aveva il valore 35.6, verrebbe visualizzato (immaginate degli spazi, ovviamente invisibili, al posto dei segni +):

++++37

Il numero è stato arrotondato, e gli sono stati aggiunti 4 spazi in testa, in modo che assieme alle 2 cifre fanno un totale di 6 caratteri, ossia esattamente il numero dei simboli # che erano stati usati nell'istruzione PRINT USING. Se ne aveste usati sette, gli spazi aggiunti (in questo caso) sarebbero stati 5.

Se il valore originario di NU avesse comportato più di 6 cifre intere, il computer avrebbe visualizzato 6 asterischi invece del numero, a segnalare che il numero risultava troppo lungo per il formato prescelto.

Lo stesso tipo di istruzione può essere impiegato per definire il numero di caratteri con cui si vuole stampare una stringa. Per esempio, battendo

PRINT USING"####";"ABCDEFG"

verrebbero visualizzati solo i caratteri ABCD, dato che si è detto al computer di visualizzare solo 4 caratteri.

Si può anche specificare al computer di indicare se un numero è positivo o negativo, e dove porre il segno + o -. Per esempio, battendo

PRINT USING"+####";4325

il computer visualizzerà 4325 preceduto dal segno +. Se il numero fornito fosse stato negativo, sarebbe stato indicato il segno - davanti al numero (fate la prova con -4325 o altro numero negativo).

Se invece battete

PRINT USING"####+";4325

questa volta il segno + verrà posto dopo il numero 4325.

Se provate ad inserire il segno - nei due esempi precedenti, ossia provate le due linee

PRINT USING"—####";4325

PRINT USING"####—";4325

usando numeri positivi e negativi, vedrete che in questi casi (con il —entro il "formato" di stampa) verrà visualizzato il segno - prima o dopo il numero, se questo è effettivamente negativo; ma se esso è positivo non verrà visualizzato il segno +.

Ma c'è di più: è anche possibile specificare al computer il numero di cifre prima e dopo il punto decimale che deve visualizzare nella stampa di numeri. Così ad esempio battendo

PRINT USING"####.####";143.65786

vedrete che il computer arrotonderà il numero a 143.6579, con quattro cifre decimali, e metterà due spazi (qui simboleggiati col +) davanti al numero, così:

++143.6579

Infatti abiamo detto al computer che volevamo 4 cifre dopo il punto decimale, per cui ha dovuto procedere ad un certo arrotondamento; e di prevedere 5 cifre davanti al punto decimale, per cui, essendo 3 le cifre intere, esso ha aggiunto due spazi davanti.

Torna talvolta comodo “spezzare” un lungo numero con l’inserimento di virgole (N.d.T.: secondo l’uso anglosassone, là dove noi useremmo dei punti, che con la notazione dei computer si confonderebbero però col punto decimale), per maggiore chiarezza di lettura. Così, scrivere 1,000,000 rende la lettura più facile che non 1000000. Anche qui possiamo impiegare il PRINT USING per dire al computer di disporre le virgole nei punti in cui torna comodo. Provate con questo esempio:

PRINT USING"##,###,###";1000000

e vedrete comparire sullo schermo

+1,000,000

Se cambiate il numero in 1000 vedrete apparire

+++++1,000

Infatti si è detto al computer di visualizzare un totale di 7 cifre, escluse le virgole (ecco perchè sono stati aggiunti 5 spazi), e di disporre le virgole ogni gruppo di tre cifre dalla sinistra (per 1,000 la virgola sta prima della 3.a cifra da destra, e per 1,000,000 prima della 3.a e 6.a cifra da destra).

Il PRINT USING torna utile specie quando si devono stampare importi di denaro. Provate questo esempio:

PRINT USING"$#######",;1234

Sullo schermo vedrete

$+++1234

Se però battete

PRINT USING"#$######";1234

il risultato sarà (più efficacemente)

+++$1234

Se si dispone il segno del $ dopo il primo simbolo #, si specifica al computer che deve considerarlo “mobile”, ossia che dovrà posizionarlo in ogni caso davanti alla prima cifra significativa dell’importo. Il numero complessivo (incluso quindi il primo) degli # è sempre quello considerato

valido dal computer per il calcolo del numero di cifre massime: così nell'esempio considerato, dove c'era un # prima e 6 dopo il segno $, il numero di cifre massime è 7. Se il segno del $ è posto prima del primo #, allora esso viene visualizzato all'estrema sinistra del numero, ossia davanti agli eventuali spazi che venissero aggiunti.

Se dovesse essere necessario visualizzare un numero in notazione "scientifica", per es. 3E+04 oppure 7.0E-02, ad indicare rispettivamente 30000 o 0.07, il comando PRINT USING è ancora utilizzabile. Nel primo caso ad es. basta aggiungere quattro simboli ↑ dopo il PRINT USING, così:

PRINT USING"#↑↑↑↑";14326

che fornirà

1E+04

Si noti come il computer ha arrotondato 14326 a 10000. Con

PRINT USING"##↑↑↑↑";14326

si ottiene invece

14E+03

e questa volta il valore arrotondato è 14000.

Un'altra caratteristica di PRINT USING che può tornare utile è la possibilità di centrare una stringa in un dato campo. Se in una certa colonna di una tabella sono previsti 6 caratteri, e volete che le lettere AA appaiano centrate in essa, il computer dovrebbe stampare due spazi, le due A, ed altri due spazi. Provate allora così:

PRINT USING"######";"AA"

ed il computer visualizzerà

++AA++

senza naturalmente che gli spazi (simboleggiati dai +) siano visibili. Dopo PRINT USING si sono indicati sei simboli #, a specificare al computer che il campo doveva essere largo 6 caratteri. Il simbolo di = posto in fondo specifica inoltre che i caratteri debbono apparire centrati.

Infine, se volete "giustificare" completamente a destra una stringa in un certo campo, si usa un'istruzione del tipo

PRINT USING"#####>";"AA"

che produce

+++AA

In questo caso si è indicato al computer di visualizzare i caratteri AA giustificati a destra in un campo largo 5 caratteri.

Sommario

L'istruzione PRINT USING può essere impiegata in diversi modi per facilitare la visualizzazione di tabelle, ed in genere per dare un aspetto ordinato allo schermo. Le varie possibilità sono:

1) definire il numero di caratteri o cifre che si vogliono visualizzare in un dato campo (un simbolo # corrisponde ad un carattere)

2) definire se si desidera un segno più o meno prima o dopo un numero (ponendo un + o - prima o dopo i simboli #)

3) specificare quante cifre si vogliono prima e dopo il punto decimale (disponendo il . al giusto posto nella serie di #)

4) specificare se si vogliono inserite delle virgole di separazione (disponendole al giusto posto nella serie di #)

5) indicare al computer di porre un simbolo $ (od altro) prima di un numero, in posizione fissa o mobile (inserendo il $ prima o dopo la serie di #, oppure dopo il primo #)

6) visualizzare un numero in forma esponenziale ("scientifica"), aggiungendo ↑↑↑↑ dopo la serie di #

7) centrare una stringa in un campo, usando il simbolo = dopo la serie di #

8) visualizzare una stringa giustificata a destra in un campo, usando il simbolo > dopo la serie di #

Naturalmente, in tutti i punti in cui nei vari esempi abbiamo usato numeri

e stringhe, si potevano usare al loro posto variabili numeriche e stringa equivalenti; e viceversa dove sono state usate variabili si possono usare numeri o stringhe. L'istruzione PRINT USING si può usare sia in modo diretto che da programma.

PUDEF

Il comando PUDEF serve per cambiare il modo con cui PRINT USING visualizza numeri e stringhe. Per esempio, se volete che il computer visualizzi degli asterischi al posto degli spazi previsti dal PRINT USING, usate questa istruzione:

PUDEF"*"

Così, se era prevista l'istruzione

PRINT USING"#####";12

il computer visualizzerà il risultato nella forma

***12. In certi casi, potete voler inserire un trattino invece di una virgola: allora si farà

PUDEF" -" (notare lo spazio prima del -)

Oppure, se desiderate un punto invece della virgola, potete usare

PUDEF" ."

Od ancora, potreste voler stampare un simbolo / invece del punto decimale, ed allora lo modificherete con

PUDEF" ,/"

In altri casi può essere utile sostituire il simbolo del dollaro con quello della sterlina:

PUDEF" ,.£"

Col comando PUDEF potete modificare uno qualsiasi fra i simboli , . $, nonchè "spazio", in un altro di vostra scelta. Avrete infatti già notato come nel comando PUDEF il primo carattere è quello del simbolo che sostituisce lo spazio; il secondo quello che sostituisce la virgola; il terzo

quello che sostituisce il punto decimale, ed infine il quarto sostituisce il simbolo $. Quindi, per rimettere tutte le cose come all'origine, si farà

PUDEF" ,.$"

Se invece volete cambiare il punto decimale in % si farà

PUDEF" ,%"

con il simbolo % al posto solitamente occupato dal punto.

La grafica

Sinora i vostri programmi avevano certi limiti, in quanto essi non sfruttavano al completo le eccellenti posssibilità di colore e grafica offerte dal vostro computer. Saprete già probabilmente che esso è in grado di visualizzare 121 colori diversi, e di creare delle immagini veramente buone: è dunque giunto il momento di mostrarvi come si fa.

COLOR

Non è il titolo (sbagliato) di questo capitoletto, ma un nuovo comando del vostro computer che si utilizza per cambiare il colore dello schermo, dello sfondo e di quanto è visualizzato. Se battete

COLOR 0,15,6

l'intero schermo assumerà un solo colore.

Il primo dei numeri posti dopo il comando COLOR indica al computer l'"oggetto" del cambiamento di colore. I suoi valori vanno da 0 a 5, e significano rispettivamente:

0 - COLORE DELLO SCHERMO
1 - COLORE DEL TESTO
2 - COLORE MULTIPLO 1 (MULTI-COLOR 1)
3 - COLORE MULTIPLO 2 (MULTI-COLOR 2)
4 - COLORE DEL BORDO

Fino a questo punto non abbiamo mai incontrato la "modalità colore multiplo (MULTICOLOR)", ma quando la affronteremo ricordatevi che il comando COLOR serve a selezionare i colori.

Il secondo numero specifica il colore desiderato. Il suo valore va da 1 a 16: 1 vale per nero, 2 bianco, 3 rosso, e così via. Il numero di ciascun colore è quello indicato sul tasto dove è scritto il relativo colore. I numeri per i colori "inferiori" di ciascuno tasto si hanno sommando 8 al valore del tasto (così ad es. il colore rosa ha numero 12, ossia 4+8).

Il terzo numero indica la "brillantezza", ossia il grado di luminosità. Il valore varia fra 0 e 7, con 0 per il livello più scuro, e 7 per quello più luminoso. Sul colore nero la luminosità non agisce. La possibilità di variare la luminosità dei vari colori accresce il loro numero apparente, per cui con 16 colori disponibili se ne possono ottenere in pratica 121.

Per poter vedere tutta la gamma di colori di cui dispone il vostro computer, battete e fate girare questo programmino:

```
10 SCNCLR
20 FORC=1TO16:FORI=0TO7
30 COLOR1,C,I:PRINT"[RVS ON] [SPAZIO] [RVS
    OFF]";
40 NEXTI,C
```

Questo programma fa variare il colore del testo attraverso tutte le possibili combinazioni di colore e di luminosità, visualizzando un quadratino colorato per ciascuna di queste combinazioni. Se volete vedere anche lo schermo ed il bordo nelle possibili colorazioni, cambiate la linea 30 in

30COLOR0,C,I:COLOR4,C,I:FOR M=1 TO 500:NEXT M

GRAPHIC

Il comando GRAPHIC serve a selezionare il modo grafico prescelto. Esistono cinque diversi modi grafici, ciascuno con certi vantaggi e svantaggi. Essi sono:

GRAPHIC 0: schermo normale con testo

GRAPHIC 1: schermo normale ad alta risoluzione con 200 righe e 320 colonne (=200x320 pixel)

GRAPHIC 2: simile al precedente, salvo che esistono 5 righe sul fondo per il testo. Lo schermo ad alta risoluzione comprende ora 180x320 pixel

GRAPHIC 3: grafica multi-colore. Consente di avere più colori in una stessa area; la risoluzione è di 200x160 pixel

GRAPHIC 4: simile al precedente, salvo per la presenza di 5 righe di fondo per il testo. La risoluzione è di 180x160 pixel

La forma tipica di un comando GRAPHIC è

10 GRAPHIC 2,1

L'aggiunta dell'1 alla fine ha l'effetto di selezionare il modo grafico e di o avete cancellare lo schermo. Tralasciando l'1 finale o ponendovi terminato di usare la grafica potete recuperare questa memoria battendone il modo grafico prescelto, ma senza cancellare lo schermo.

La modalità grafica comporta un grande consumo di memoria, ma quando avete finito di usarla, potete accedere di nuovo alla memoria battendo

GRAPHIC CLR

che vi permette di riaccedere alla parte di memoria che il computer riserva quando viene usata la grafica.

LOCATE

Lo schermo ad alta risoluzione è dotato di uno speciale tipo di cursore, detto "cursore pixel", che risulta in pratica invisibile, ma indica al computer il punto esatto in cui si sta disegnando sullo schermo. Anche se non lo vediamo, lo possiamo muovere in una posizione qualsiasi dello schermo: per fare questo, dobbiamo indicare al computer di quanti pixel si deve spostare il cursore pixel in orizzontale ed in verticale, così:

175 LOCATE 40,30

Questa istruzione specifica al computer che deve posizionare (LOCATE= collocare) il cursore pixel nella posizione di coordinate 40 (dal bordo sinistro dello schermo) e 30 (dall'alto dello schermo), in unità pixel.

Possiamo anche dire al computer di spostare il cursore pixel di un certo numero di pixel in su, in giù, a sinistra od a destra, così:

240 LOCATE +10,—5

che indica al computer che deve spostare il cursore pixel di 10 pixel verso destra e 5 pixel verso l'alto, rispetto alla posizione attuale. Se avessimo

voluto uno spostamento di 10 unità verso sinistra e 5 verso il basso, avremmo usato

255 LOCATE —10,+5

Si può anche istruire il computer a spostare il pixel di un certo numero di unità secondo un certo angolo: questo è misurato in gradi, rispetto alla verticale ("NORD" dello schermo):

354 LOCATE 40;45

sposta il cursore pixel di una distanza di 50 pixel, formando un angolo con la verticale di 45 gradi ("in direzione NE")

DRAW

Il comando DRAW permette di disegnare sullo schermo. Si può tracciare un singolo punto, oppure una linea diritta, o più linee . Provate con questo programmino:

```
10 GRAPHIC1,1:COLOR 0,1,7:COLOR 4,7,7:COLOR 1,2,7
20 DRAW 1,10,10
```

Se date il RUN, vedrete comparire sullo schermo (che appare nero con un bordo blu) un punto bianco.

Con la linea 10 si seleziona il modo grafico n.o 1, e si cancella lo schermo, assegnando poi a questo il colore nero, al bordo il colore giallo, ed al disegno il colore bianco.

Se poi esaminate la linea 20, vedrete che dopo il comando DRAW compaiono tre numeri. Il primo è il colore., che può essere un valore fra 0 e 3. Se si sceglie lo 0 il disegno avviene nel colore dello sfondo, il che provoca la cancellazione di ciò su cui la traccia passa una seconda volta. Se si sceglie il valore 1 il colore del disegno sarà quello selezionato per il testo (nel nostro caso, il bianco). Se si sceglie 2, il disegno verrà eseguito in multi-colore 1, e con il valore 3 in multi-colore 2. Dato che questo programma non usa le modalità multi-colorate, la selezione di un colore 2 o 3 per DRAW fa sì che il disegno venga eseguito nel colore che è stato in precedenza assegnato a tale fonte (p. es. se il multi-colore 1 aveva ricevuto assegnato il 3, il computer disegnerà con una traccia rossa).

Il secondo dei tre numeri dopo DRAW specifica la coordinata orizzonta-

le, ed il terzo la coordinata verticale. Con la linea 20 si dice quindi al computer di partire con la traccia dal punto di coordinate (10,10), e dato che per ora non gli si fa disegnare nulla, esso si limiterà a tracciare un singolo punto in tale posizione.

Se esaminate questo punto sullo schermo potrete vedere quanto è piccolo un pixel. Se pensate che (per lo meno in modo GRAPHIC 1) ci sono 320 colonne e 200 righe di pixel sullo schermo, potete rendervi conto del grado di dettaglio che si può così raggiungere.

Il pixel che compare sullo schermo vi rimarrà visibile sino a quando direte al computer di ritornare al normale schermo per i testi. Il modo più comodo per fare questo è di battere un carattere qualsiasi (diverso da un numero) e premere RETURN. Lo schermo tornerà normale, ed anche se comparirà un messaggio di errore in realtà non si è compromesso nulla.

Se adesso cambiate la linea 20 in

```
20 DRAW 1,0,0 TO 319,199
```

e date il RUN, vedrete disegnare una linea obliqua che attraversa lo schermo dal pixel (0,0) a quello opposto di coordinate (320,200). E si può continuare con altri comandi DRAW come ad es.

```
20 DRAW 1,0, TO 319,199 TO 319,0 TO 0,199 TO 0,0
```

che disegnerà due triangoli con un vertice in comune al centro dello schermo.

Si può anche ordinare al computer di disegnare una linea dalla posizione corrente del cursore pixel ad un altro punto qualsiasi dello schermo. Basta fare, per es.,

```
20 LOCATE 10,10:DRAW 1 TO 50,50
```

Dando il RUN, il cursore pixel si posizionerà al punto (10, 10), e poi verrà tracciata la linea che congiunge tale punto a quello di coordinate (50,50), nel colore corrente del testo.

Si può anche dire al computer di tracciare un segmento da un dato punto ad un altro posto a un certo numero di pixel di distanza a sinistra o a destra, ed a tanti pixel più su o più giù del punto di partenza, in questo modo:

```
20 LOCATE 10,10:DRAW 1 TO +20,+10 TO -10,-5
```

Dando il RUN al programma dopo quest'ultima modifica si vedrà tracciare una linea dal punto dove era inizialmente il cursore (di coordinate (10,10)) sino ad un punto 20 pixel più a destra e 10 pixel più in basso, e poi ancora una linea da questo nuovo punto ad uno posto 10 pixel a sinistra e 5 pixel più su.

Il comando DRAW può anche servire a disegnare linee sotto un certo angolo. Cambiando la linea 20 ancora in

```
20 LOCATE 50,50:DRAW 1 TO 40;32
```

e dando il RUN, verrà tracciata una linea dal punto (50,50) ad una distanza di 40 pixel, sotto un angolo di 32 gradi rispetto alla verticale per il "NORD".

BOX

Il comando BOX serve per disegnare rapidamente quadrati o rettangoli. Per vedere come, cambiate la solita linea 20 precedente in

```
20 BOX 1,110,50,210,150
```

e date il RUN: vedrete apparire un quadrato al centro dello schermo. Con tale istruzione si dice al computer di tracciare un rettangolo (BOX=(letteralmente) riquadro) il cui vertice sinistro in alto è nel punto (110,50), ed il vertice opposto, a destra in basso, nel punto (210,150). Il computer calcola automaticamente la posizione degli altri due vertici, e provvede poi a tracciare le quattro linee che li congiungono. (Ricordiamo inoltre che il primo dei 5 numeri dopo BOX fissa la modalità di colore, come nel caso di DRAW (vedi)).

Risulta anche possibile effettuare la rotazione di un rettangolo. Se ad es. aggiungete ',45' alla fine dell'istruzione 20 di cui sopra,

```
20 BOX 1,110,50,210,150,45
```

e date il RUN al programma, vedrete che viene disegnato un quadrato che appare ruotato di 45 gradi rispetto alla posizione originale. L'angolo può avere un valore qualsiasi fra 0 e 360 gradi (come si sa, 0 e 360 gradi corrispondono, ed entrambi non provocano alcuna rotazione!).

Potete anche colorare l'interno del rettangolo in un colore assegnato:

basterà porre ',1' dopo il valore dell'angolo di rotazione in gradi (od anche ',1' semplicemente, se non si vuole alcuna rotazione), come in

20 BOX 1,110,50,210,150,45,1

oppure

20 BOX 1,110,50,210,150,,1

L'1 in fondo dice al computer di colorare il rettangolo del colore attualmente valido per i testi (ossia, il colore con cui è stato disegnato il contorno del rettangolo).

CIRCLE

Come avrete probabilmente già indovinato, con il comando CIRCLE si possono tracciare dei circoli, ma inoltre si possono anche disegnare ellissi, triangoli, quadrati, e in pratica quasi tutte le forme di figure chiuse, come vedremo subito.

Modificate la linea 20 del solito programma in

20CIRCLE 1,160,100,80

Facendo eseguire il programma vedrete che viene disegnato un circolo di raggio 80 pixel, con il centro nella posizione centrale dello schermo.

Il primo numero definisce al solito il colore (al modo di DRAW), ed il secondo ed il terzo le coordinate del centro. Il quarto dà il valore da assegnare al raggio del circolo.

Se ora aggiungete a questa linea, in fondo, ',4', ossia

20 CIRCLE 1,160,100,80,40

e date il RUN, vedrete tracciare un'ellisse, con l'asse minore (verticale) metà di quello maggiore. Infatti in genere i numeri da indicare sarebbero due, uno la misura del semi-asse orizzontale, l'altro quella del semi-asse verticale (sempre in pixel). Quando si omette il secondo dei due valori si assume automaticamente che essi sono eguali, per cui viene tracciato un circolo. Naturalmente se il semiasse verticale risulta maggiore di quello orizzontale (indicato per primo) l'ellisse assume una forma allungata verticalmente.

È anche possibile tracciare archi di circolo. Alterando come segue la linea:

20 CIRCLE 1,160,100,80,80,90,180

si dice al computer di tracciare, per il circolo di centro (160,100) e raggio 80, solo l'arco che va dall'angolo di 90 gradi (6° numero dopo CIRCLE) all'angolo di 180 (7° numero).

Si può anche fare ruotare il dato circolo od arco (anche se a prima vista possa apparire inutile: ma vedremo tosto che non è così). Basterà aggiungere il valore dell'angolo di rotazione in gradi come ultimo parametro della lunga lista (l'8.o), così

20 CIRCLE 1,160,100,80,80,90,180,20

Questo programma disegnerà ancora un arco di 90 gradi, ma ruotato di 20 rispetto alla posizione originale.

Infine, come detto, potete tramite CIRCLE disegnare altre figure chiuse, come triangoli od esagoni Per disegnare un poligono, bisogna specificare al computer di quanti gradi deve ruotare per tracciare il lato successivo: questo valore si ricava da 360 diviso per il numero di lati. Cosi ad esempio se vogliamo tracciare un triangolo equilatero, basta aggiungere ',120' in fondo alla linea 20, così:

20 CIRCLE 1,160,10,80,80,0,0,0,120

SCALE

Questo SCALE è un comando relativamente semplice: esso può risultare attivo o disattivato. Per attivarlo, basta fare SCALE 1, e per disattivarlo SCALE 0. Se al solito programma usato sin qui aggiungiamo

15 SCALE 1

e si dà il RUN, vedremo apparire il triangolo (supponendo di lavorare sull'ultima fra le tante versioni sperimentate del programmino...) nell'angolo superiore sinistro dello schermo, e assai più piccolo. Il comando SCALE infatti fa supporre al computer che lo schermo si componga di 1024x1024 pixel, anche se in realtà esso continua ad avere le reali dimensioni di 320x200 pixel (in alta risoluzione completa), opure 160x200 (in modo MULTICOLOR). La cosa può risultare utile per certi grafici.

PAINT

Il comando PAINT serve per "riempire di colore" alcune parti delle vostre figure. Si tratta di un comando semplice da usare: prima di tutto dovete indicare al computer il colore prescelto (al modo DRAW, come al solito), e quindi da che punto deve cominciare a colorare. Cancellate quindi la linea 15, date in modo diretto il comando SCALE 0, e poi battete questa linea:

30 PAINT 1,160,100,1

e vedrete che il vostro triangolo viene riempito di colore bianco. Con questa istruzione si è detto al computer di riempire di colore (colore del testo) una certa area, iniziando dal punto di coordinate 160,100. L'1 finale specifica al computer che deve interrompere la colorazione se incontra nel suo cammino una qualsiasi linea disegnata in un colore diverso da quello dello sfondo. Se si tralascia l'1, la colorazione continuerà invece sino a quando si incontrerà una linea di confine dello stesso colore che si usa per colorare l'interno della figura.

Per selezionare il colore di PAINT bisogna se del caso modificare il colore corrente del testo in quello in cui si vuole colorare. Per esempio, se si vuole colorare l'interno del triangolo in rosso, occorre cambiare il colore del testo, così:

25 COLOR 1,3,4

Grafica a più colori

Se avete sperimentato un po' con la grafica vi sarete accorti che in una certa posizione di stampa o quadratino non si possono avere più di due diversi colori (ricordate la suddivisione dello schermo in posizioni di stampa? Se avete dei dubbi, tornate a riguardare il capitoletto che trattava del comando CHAR). Se impostate questo breve programma

```
10 GRAPHIC1,1:COLOR0,1,7:COLOR4,1,7:COLOR1
   ,2,7
20 CIRCLE1,50,50,20:PAINT1,50,50
30 COLOR1,5,5
40 CIRCLE1,70,70,25:PAINT1,70,70
50 COLOR1,6,5
60 CIRCLE1,40,90,15:PAINT1,40,90
```

vedrete che sullo schermo vengono disegnati alcuni cerchi che poi vengono colorati internamente. Ma vi accorgerete pure che dove i circoli si sovrappongono, degli interi quadratini cambiano colore, e questo succede perchè non si possono avere più di due diversi colori in uno stesso quadratino, uno dei quali è il colore dello sfondo. Per superare questa difficoltà si deve ricorrere alla modalità MULTICOLOR. Modificate il vostro programma a questo modo:

```
10 GRAPHIC3,1
15 COLOR0,1,7:COLOR4,1,7:COLOR1,2,7
20 CIRCLE1,50,50,20:PAINT1,50,50,1
30 COLOR2,5,5
40 CIRCLE2,70,70,25:PAINT2,70,70,1
50 COLOR3,6,5
60 CIRCLE3,40,90,15:PAINT3,40,90
```

Se date il RUN a questo programma, vedrete che questa volta vengono disegnati tre cerchi, colorati internamente, e ciascuno mantiene il proprio colore, senza strane contaminazioni! Dipende dal fatto che lavoriamo in modo multi-colorato, in cui possiamo avere sino a quattro distinti colori in uno stesso quadratino (dei quali uno è sempre il colore dello sfondo).

Avrete notato che abiamo usato un colore diverso per ciascun circolo, ognuno definito da apposito comando COLOR. Il secondo circolo è disegnato in MULTICOLOR 1 (colore sorgente 2), ed il terzo in MULTICOLOR 2 (colore sorgente 3). Il primo circolo è sempre disegnato in colore sorgente 1 (normale colore per i testi).

Vi accorgerete tuttavia che i vari cerchi appaiono un pò più "grossolani": ciò dipende dal fatto che in modo MULTICOLOR la risoluzione orizzontale si dimezza, per cui invece di 320 pixel ne abbiamo solo 160 in ogni linea. Perciò per figure in forte dettaglio che non richiedono un'eccessivo numero di colori, è meglio adottare GRAPHIC 1 o GRAPHIC 2, ma se il colore assume maggiore rilievo e varietà è meglio usare GRAPHIC 3 oppure GRAPHIC 4.

Una particolarità ottenibile col colore sorgente 3 quando si è in modalità MULTICOLOR è che risulta allora possibile modificare il colore di qualsiasi cosa sia stata disegnata usando tale colore. Se per esempio avete tracciato un cerchio usando il colore sorgente 3 bianco, il circolo sarà bianco. Se poi modificaste il colore sorgente 3 in rosso, il circolo appena tracciato cambierebbe immediatamente il suo colore in rosso: la cosa non

si verifica per gli altri colori sorgenti. Provate ad aggiungere al programma precedente queste linee:

```
70 FOR C=1 TO 8:FOR I=0 TO 7
80 COLOR 3,C,I:FOR M=1 TO 5OO:NEXT M
90 NEXT I,C
```

Se date il RUN al programma vedrete ora che uno dei circoli (quello disegnato col colore sorgente 3) cambia continuamente di colore passando attraverso tutte le combinazioni di colore e luminosità previste dal vostro computer.

SSHAPE e GSHAPE

Si tratta di due comandi per la grafica molto utili, perchè consentono di memorizzare una certa area dello schermo in una variabile stringa, e spostarla poi in un'altra zona a piacere. Il che significa che in questo modo è possibile muovere una forma abbastanza grande e complessa (p. es. un'astronave) in maniera abbastanza "fluida" attraverso lo schermo.

Il comando SSHAPE serve appunto a memorizzare in una variabile stringa una parte dello schermo. Tutto quello che serve è specificare al computer in quale variabile stringa si vuole memorizzare la forma, ed indicare le coordinate dell'angolino superiore sinistro e dell'angolo inferiore destro della figura. Eccovi un esempio:

```
320 SSHAPE SP$, 40,50,60,70
```

In questo modo si dice al computer di memorizzare entro la variabile SP$ la figura compresa fra il punto di coordinate (40,50) come vertice superiore sinistro, e il punto (60,70) come vertice inferiore destro.

Se volete riportare sullo schermo, in una certa posizione, questa figura, basterà indicare la variabile stringa in cui è memorizzata, e le coordinate in cui si deve posizionare il vertice superiore sinistro della figura. L'istruzione

```
400 GSHAPE SP$, 60,70
```

fa comparire sullo schermo la forma che era stata memorizzata in SP$, con il suo vertice superiore sinistro posto nel pixel di coordinate (60,70). Battendo questo corto programma e facendolo girare vedrete una "palla"

che si muove lentamente attraverso lo schermo e rimbalza sui suoi bordi (purtroppo, con questo metodo i movimenti sono molto lenti)

```
 10 COLOR1,6,5:COLOR0,2,7:COLOR4,3,4:GRAPHI
    C1,1
 30 CIRCLE1,160,100,3:PAINT1,160,100
 40 SSHAPEBALL$,156,96,164,104
 50 X=160:Y=100:XD=1:YD=1
 60 GSHAPEBALL$,X-4,Y-4
 70 X=X+XD:Y=Y+YD
 80 IFX>315THENXD=-1
 90 IFX<4THENXD=1
100 IFY>195THENYD=-1
110 IFY<4THENYD=1
120 GOTO60
```

Ed ecco le spiegazioni sul funzionamento del programma:

Linea 10: fissa il colore DRAW a verde, quello dello sfondo bianco, e quello del bordo rosso; seleziona il modo grafico 1 e cancella lo schermo
Linea 30: disegna nel colore corrente del testo un circolo con raggio di 3 pixel e centro in (160,100), che poi colora in colore del testo
Linea 40: memorizza la forma che ha il suo vertice sinistro in alto al punto (156,96) e il vertice inferiore destro in (164,104) nella variabile stringa BALL$
Linea 50: assegna il valore 160 alla variabile X, 100 alla variabile Y e 1 alle variabili XD e YD
Linea 60: posiziona la figura memorizzata in BALL$ con il vertice sinistro in alto posizionato sul punto di coordinate (X-4,Y-4)
Linea 70: aggiungi il valore di XD alla variabile X **Linea 80**: se il valore di X supera 315 assegna il valore -1 alla variabile XD
Linea 90: se il valore di X risulta inferiore a 4 assegna il valore 1 alla variabile XD
Linea 100: se il valore di Y supera 195, assegna il valore -1 alla variabile YD
Linea 110: se il valore di Y è minore di 4 assegna il valore 1 alla variabile YD
Linea 120: salta alla linea 60 e riprendi l'esecuzione da quel punto

Per ora, ci limitiamo a "riportare" la figura che abbiamo memorizzato in BALL$ esattamente come l'abbiamo "prelevata". Naturalmente potrem-

mo "riprenderla" modificando qualcosa. Per fare un esempio, provate a cambiare la linea 60 in

60 GSHAPE BALL$,X-4,Y-4,1

Se date il RUN vedrete stavolta la palla muoversi come prima attraverso lo schermo, però invertita: vale a dire che mentre prima la palla era verde su uno sfondo bianco, ora è bianca su uno sfondo verde.

Modificando ancora la linea 60 in

60GSHAPE BALL$,X-4,Y-4,2

dopo il RUN vedrete una linea verde con un fronte curvo che si sposta attraverso lo schermo. Si tratta ancora della palla, ma stavolta il contorno della stessa ha subito un'operazione di OR con lo sfondo. Il che significa che il computer posiziona la figura sullo schermo, e ne paragona ogni punto con il punto dello schermo che andrà ad occupare. Se uno solo, od entrambi, questi due punti sono "illuminati", il punto sullo schermo risulterà illuminato. Il che significa pure che si possono porre due figure una sopra l'altra, e far sembrare che l'una si muova sopra l'altra. Così se per es. si esegue l'OR fra la forma '|' e la '—' il risultato è una '+'!

Provate ora a modificare ancora la linea 60 come segue, e vederne l'effetto:

60 GSHAPE BALL$,X-4,Y-4,3

Questa volta dopo il RUN vedrete ... sparire la palla, che non ricompare più! Se però aggiungete la linea

20 COLOR 1,3,4:BOX 1,0,150,319,170,1:COLOR 1,6,5

e date nuovamente il RUN, vedrete una striscia rossa tracciata nella parte inferiore dello schermo: quando la palla "invisibile" la attraversa, vedrete che parte della striscia rossa si muta in verde, e formarsi un percorso in bianco che traversa la striscia. Questa volta si è eseguita un'operazione di AND fra la palla e lo sfondo. Il che significa che quando il computer posiziona la figura sullo schermo, paragona ogni suo punto con quello in cui andrebbe piazzato, e rende luminoso quest'ultimo solo se ambedue i punti risultano illuminati. Per esempio, l'operazione di AND fra la figura '+' e la '|' porta alla '|', perchè solo questa parte risulta comune alle due figure.

Infine, se modificate la linea 60 in
60 GSHAPE BALL$,X-4,Y-4,4

e date il RUN vedrete la palla muoversi sulo schermo lasciando una strana traccia dietro di sè. In questo caso si è eseguita l'operazione XOR (OR esclusivo) fra i punti della figura e quelli dello sfondo: resta illuminato solo quel punto dello schermo in cui solo una delle due figure ha un punto luminoso, ma non entrambe. Il che significa che lo XOR fra '|' e '—' dà come risultato '+' (col pixel centrale "spento"), ma lo XOR di '|' con '|' dà ' '.

RCLR

Si tratta di una funzione, che viene impiegata per determinare quale colore è assegnato ad un certo colore sorgente. Se per esempio abbiamo in un programma la linea

320 Z=RCLR(0)

alla variabile Z viene assegnato il valore del colore corrente dello sfondo. Il numero fra parentesi va da 0 a 4, proprio come nel caso del comando COLOR.

RLUM

Anche questa è una funzione, che fornisce come risultato il livello corrente di luminosità di un colore sorgente. Viene usato in modo simile a RCLR, ossia con la linea

1550 C2=RLUM(4)

alla variabile C viene assegnato il valore corrente della luminosità del bordo.

RGR

Ancora una funzione grafica, che serve a determinare quale è il modo grafico corrente. RGR si usa come in

30 F=RGR(0)

dove alla variabile F viene assegnato il valore (0-4) del modo grafico corrente. Il numero fra parentesi non ha significato, e può assumere un valore qualsiasi (conviene lo 0, tanto il computer lo trascura).

RDOT

Altra funzione grafica, che fornisce informazioni sulla posizione del cursore pixel. Viene usata come negli esempi che seguono:

100 Z=RDOT(0)

assegna a Z il valore della coordinata X corrente del cursore pixel;

100 PRINT RDOT(1)

visualizza il valore della coordinata Y corrente del cursore pixel; e

100 AA=RDOT(2)

assegna il valore del colore sorgente corrente alla variabile AA.

Nelle pagine successive potete trovare il listato del programma "Artista", che vi permetterà di disegnare delle figure a piacere sullo schermo, in ogni modo grafico. Sarebbe bene che lo trascriviate in memoria, e leggiate con cura le spiegazioni sul suo funzionamento, anche se solo per capire bene tutti i vari comandi grafici.

Artista

Questo programma fa largo uso delle capacità grafiche del vostro computer, permettendovi di disegnare figure sullo schermo. Potete disegnare circoli, triangoli, rettangoli, segmenti, linee punteggiate, insomma quasi ogni cosa che vi venga in mente.

Dopo aver dato il RUN al programma, vi verrà richiesto che modo grafico volete scegliere. Dovrete rispondere con un numero fra 1 e 4. Lo schermo si cancellerà, per dare uno schermo nero con un bordo nero, ed una croce rossa al centro.

Questa costituisce il vostro cursore, che potete spostare lungo lo schermo usando i tasti di controllo del cursore (frecce). Se volete, potete cambiare il colore dello schermo, premendo S seguito dal numero (da 1 a 8) del colore scelto. Allora lo schermo verrà cancellato (per cui, non fate così se volete conservare una figura già tracciata) e passerà al colore desiderato. Potete cambiare anche il colore del bordo, semplicemente premendo E seguito dal numero del colore desiderato.

Anche il colore della traccia può essere modificato (cambia allora anche il colore del cursore). Premete in tal caso la X seguita dal numero del colore scelto (che verrà assunto anche dal cursore). Se volete cambiare la luminosità, premete il tasto I seguito dal valore della luminosità desiderata: l'effetto si produrrà tuttavia solo quando modificherete il colore di qualcosa. Per esempio, una volta modificato in 7 il valore della luminosità, se cambiate a questo punto il colore del bordo in rosso, questo sarà di un bel rosso acceso, ma ogni altra cosa manterrà la luminosità originale.

La tecnica più semplice prevista in *Artista* è quella per *Disegnare.* Se premete D e fate muovere il cursore, vedrete che esso lascia una traccia sullo schermo mentre si sposta. Una seconda pressione di D disattiva questa funzione.

Potete anche disegnare delle circonferenze, o un'altra qualsiasi delle figure ottenibili tramite CIRCLE. Per far questo, dovete posizionare il cursore sullo schermo nel punto in cui volete sia il centro della figura, e quindi premere C. Lo schermo si cancellerà, ma non preoccupatevi: stavolta la vostra figura non è andata persa. Vi verranno poste domande relative al "circolo", ossia i valori dei semiassi X ed Y, l'angolo iniziale, l'angolo finale, l'angolo di rotazione, ed il numero di gradi fra ciascun segmento del "circolo". Se premete semplicemente RETURN in risposta, verrà assegnato automaticamente il valore 0 al parametro (ovvero 2 per il solo caso del "n.o di gradi per ciascun segmento"). Una volta fornite tutte le risposte, vedrete disegnarsi la figura prescelta.

Se vi dà fastidio la lentezza con cui si sposta il cursore, premete il tasto F: il cursore si sposterà allora di 5 pixel alla volta invece di 1. Premendo F una seconda volta si torna alle condizioni originali. Se fate uso della funzione *Disegno* mentre siete in modo spostamento rapido, la linea verrà disegnata punteggiata.

Potete disegnare pure dei rettangoli. Posizionate il cursore nel punto in cui deve stare uno dei vertici (non importa quale) e premete O (per "origine"). Ivi comparirà un punto illuminato. Spostate ora il cursore sulla posizione del vertice opposto del rettangolo, e premete B: verrà disegnato il rettangolo.

La medesima funzione che fissa un'*Origine* può essere usata per unire due punti dello schermo. Se fissate il cursore in una delle due posizioni e premete O, quindi lo muovete sull'altro punto e premete J, verrà disegnato il segmento che congiunge i due punti. Spostando ancora il cursore su di un'altra posizione e ripremendo J verrà disegnata la congiungente fra l'ultimo punto (dove avete premuto J la prima volta) ed il nuovo. In

questo modo, spostando ripetutamente il cursore sullo schermo ed usando J potete tracciare le varie linee consecutive di congiunzione di una figura formata da segmenti.

Potete naturalmente pure colorare l'interno di una figura chiusa. Per questo, posizionate il cursore in un punto qualsiasi all'interno della figura che volete colorare, accertatevi di aver scelto il colore desiderato, e premete il tasto P. L'area interna si colorerà del colore prescelto.

Se state adoperando uno dei modi MULTICOLOR, potete selezionare tre colori sorgente diversi. Per cambiare il colore del MULTICOLOR 1, premete il tasto con i due-punti (:) e poi il tasto del colore prescelto (1-8). Per modificare il colore nel MULTICOLOR 2 premete il tasto del punto-e-virgola (;) e poi quello del colore scelto. Una volta fatto questo potete selezionare il colore sorgente desiderato semplicemente premendo 1 per normale colore dei testi, 2 per MULTICOLOR 1, e 3 per MULTICOLOR 2.

NOTA: Se volete usare questo programma su di un Commodore 16, dovete eliminare tutte le REM nonchè tutti gli spazi fra i vari termini che compaiono in un'istruzione. Per esempio, se vedete due linee come le

```
80  REM MEMORIZZA BLOCCO VUOTO IN BK$
90  SSHAPE BK$, X-3, Y-3, X+3, Y+3
```

dovete eliminare la linea 80, e riscrivere la 90 così:

```
90SSPEBK$,X-3,Y-3,X+3,Y+3
```

altrimenti il programma non funziona.

Inoltre, quando vedete qualcosa scritto entro parentesi quadre (p. es. come in IF A$="[CURS DES") significa che dovete a quel punto premere il tasto indicato fra le parentesi (qui, il tasto "freccia a destra").

Funzionamento del programma

Artista è un programma piuttosto lungo, e non entrerò quindi in dettagli: invece vi fornirò una sintesi di quello che fa ogni sezione, per darvi una mano a comprenderlo meglio.

Il programma gira propriamente solo sul Plus 4 per ragioni di occupazione di memoria, ma quando sarà disponibile l'espansione di memoria potrà funzionare anche sul C16.

Nella linea 10 si fissano i colori dello schermo, del bordo e del testo, poi in linea 20 si assegnano i valori di certe variabili utilizzate nel programma. La linea 30 cancella lo schermo, e vi chiede quale modo grafico scegliete: la correttezza della risposta viene controllata dalla linea 40. Se date una risposta inammissibile la richiesta viene ripetuta.

La linea 50 calcola quante colonne prevede il modo grafico prescelto, e la linea 60 assegna ad Y il numero di righe. La linea 70 seleziona il modo grafico e cancella lo schermo.

La linea 90 procede a memorizzare il contenuto dello schermo attorno al cursore (che per ora è solo un'area vuota, perchè lo schermo è stato cancellato) nella variabile BK$; le linee 110-120 disegnano il cursore. Il cursore viene memorizzato in CU$ nella linea 140, prima della nuova cancellazione dello schermo (linea 160). Poi il cursore appena memorizzato in CU$ viene posizionato sullo schermo dalla linea 180.

Le linee 190-300 servono ad effettuare la scansione della tastiera ed a saltare alle varie subroutine, che cambiano il colore dello schermo e del bordo, disegnano 'circoli', ed altre cose ancora. Gli spostamente del cursore sono effettuati con le linee 320-350.

La funzione *Disegno* viene attivata e disattivata alla linea 370, mentre la 390 fissa l'origine. La funzione *Join* (congiungimento di due punti con un segmento) è curata dalle linee 410-420, mentre con la 440 si attiva la funzione *Box* (disegno di un rettangolo). Se avete premuto la F, la linea 460 attiva o disattiva lo spostamento rapido, con l'incremento a 5 pixel invece di 1 fissato in linea 480.

È importante che il cursore non possa uscire dai limiti dello schermo, per cui le linee 500 e 510 verificano che ciò non avvenga quando si muove il cursore. La linea 530 cancella il precedente cursore prima che questo venga riportato nella nuova posizione dalla linea 560. L'area occupata dal cursore sullo schermo è memorizzata in BK$ in modo che il cursore non cancelli quanto è già presente sullo schermo.

La linea 590 traccia un punto sullo schermo se è attiva la funzione *Disegno* (oppure se è stata appena fissata l'*Origine*, che viene pure marcata da un punto). La linea 600 reinizializza le variabili Q e P se il punto appena tracciato è l'origine.

La linea 620 provvede a disegnare un rettangolo. Questa routine prima elimina il cursore, poi disegna il rettangolo e memorizza l'area che si troverà sotto il cursore (che costituisce uno dei vertici) in BK$. La linea

630 rimanda il programma alla linea 170, per ricollocare il cursore sullo schermo e rinnovare il sondaggio della tastiera.

Le linee 650-750 contengono le subroutine che servono a cambiare i colori e la luminosità. Le linee 770-860 richiedono le dimensioni per il "cerchio" e lo disegnano, e le linee 880-890 colorano l'interno di una figura nel colore prescelto. Il colore dello schermo viene modificato con le linee 910-930, mentre con le linee 940-1010 si scelgono i due MULTICOLOR. La linea 1020 sonda la tastiera ed assegna i valori numerici corrispondenti ai caratteri battuti (convertiti tramite VAL) alla variabile T.

Artista

```
 10 GRAPHIC0:COLOR0,1,7:COLOR4,1,7:COLOR1,2
    ,7
 20 C=2:I=7:S=1:Z=4:V=5
 30 SCNCLR:INPUT"QUALE MODO GRAFICO SCEGLIE
    TE";GM
 40 IFGM<1ORGM>4THEN30
 50 IFGM>2THENWD=160:X=80:ELSEWD=320:X=160
 60 Y=100
 70 GRAPHICGM,1
 80 REM MEMORIZZA BLOCCO VUOTO IN BK$
 90 SSHAPE BK$,X-3,Y-3,X+3,Y+3
100 REM DISEGNANO IL CURSORE
110 DRAW S,X-3,Y TO X-1,Y:DRAW S,X+1,Y TO X
    +3,Y
120 DRAW S,X,Y-3 TO X,Y-1:DRAW S,X,Y+1 TO X
    ,Y+3
130 REM MEMORIZZA CURSORE IN CU$
140 SSHAPE CU$,X-3,Y-3,X+3,Y+3
150 SCNCLR
160 REM METTE IL CURSORE SULLO SCHERMO
170 GSHAPE CU$,X-3,Y-3,4
180 REM ATTENDE UN COMANDO
190 GETKEY A$
200 IFA$="1"THENS=1
210 IFA$="2"THENS=2
220 IFA$="3"THENS=3
```

```
230 IFA$="X"THENGOSUB650
240 IFA$="I"THENGOSUB690
250 IFA$="E"THENGOSUB730
260 IFA$="C"THENGOSUB770
270 IFA$="P"THENGOSUB880
280 IFA$="S"THENGOSUB910
290 IFA$=":"THENGOSUB950
300 IFA$=";"THENGOSUB990
310 REM CONTROLLO DEL CURSORE
320 IFA$="[CURS DES]"THENXD=1
330 IFA$="[CURS SIN]"THENXD=-1
340 IFA$="[CURS SU]"THENYD=-1
350 IFA$="[CURS GIU]"THENYD=1
360 REM ABILITA E DISABILITA LA FUNZIONE DI
     DISEGNO
370 IFA$="D"ANDP=0THENP=1:ELSEIFA$="D"THENP
    =0
380 REM ASSEGNA ORIGINE
390 IFA$="O"THENOX=X:OY=Y:P=1:Q=1
400 REM DISEGNA UNA LINEA DALL'ORIGINE AL C
    URSORE
410 IFA$="J"THENGSHAPE BK$,X-3,Y-3:DRAW S,O
    X,OY TO X,Y:OX=X:OY=Y
420 IFA$="J"THENSSHAPE BK$,X-3,Y-3,X+3,Y+3
430 REM ABILITA LA FUNZIONE BOX
440 IFA$="B"THENB=1
450 REM ABILITA E DISABILITA LA FUNZIONE DI
     SPOSTAMENTO RAPIDO
460 IFA$="F"ANDF=0THENF=1:ELSEIFA$="F"THENF
    =0
470 REM INCREMENTO DI 5 PIXELS IN SPOSTAMEN
    TO RAPIDO ATTIVATO
480 IFF=1THENXD=XD*5:YD=YD*5
490 REM CONTROLLA CHE IL CURSORE NON ESCA D
    ALLO SCHERMO
500 IFX+XD<0ORX+XD>WDTHENXD=0
510 IFY+YD<0ORY+YD>200THENYD=0
520 REM CANCELLA IL PRECEDENTE CURSORE
530 GSHAPE BK$,X-3,Y-3
540 REM MUOVE IL CURSORE
```

```
550 X=X+XD:Y=Y+YD:XD=0:YD=0
560 REM MEMORIZZA IN BK$ L'AREA OCCUPATA DA
    L CURSORE SULLO SCHERMO
570 SSHAPE BK$,X-3,Y-3,X+3,Y+3
580 REM DISEGNA UN PUNTO SULLO SCHERMO
590 IFP=1THENGSHAPE BK$,X-3,Y-3:DRAW S,X,Y:
    SSHAPE BK$,X-3,Y-3,X+3,Y+3
600 IFQ=1THENQ=0:P=0
610 REM DISEGNA UN RETTANGOLO
620 IFB=1THENGSHAPE BK$,X-3,Y-3:BOX1,OX,OY,
    X,Y:SSHAPE BK$,X-3,Y-3,X+3,Y+3:B=0
630 GOTO170
640 REM CAMBIA IL COLORE DI DISEGNO
650 GOSUB1020:C=T
660 IFC<1ORC>8THEN650
670 COLOR1,C,I:RETURN
680 REM CAMBIA LUMINOSITA'
690 GOSUB1020:I=T
700 IFI<0ORI>7THEN690
710 COLOR1,C,I:RETURN
720 REM CAMBIA CORNICE
730 GOSUB1020:B=T
740 IFB<1ORB>8THEN730
750 COLOR4,B,I:RETURN
760 REM CHIEDE LE DIMENSIONI DEL CERCHIO
770 GRAPHIC0,1:COLOR1,2,7
780 INPUT"ASSE X";XR:INPUT"ASSE Y";YR
790 INPUT"ANGOLO INIZIALE";SA:INPUT"ANGOLO
    FINALE";EA
800 INPUT"ANGOLO DI ROTAZIONE";RA:INPUT"GRA
    DI FRA I SEGMENTI";DB
810 GRAPHICGM
820 IFDB=0THENDB=2
830 REM DISEGNA IL CERCHIO
840 COLOR1,C,I
850 CIRCLE S,X,Y,XR,YR,SA,EA,RA,DB
860 RETURN
870 REM AREA COLORATA
880 GSHAPE BK$,X-3,Y-3
```

```
 890 PAINT S,X,Y,1:SSHAPE BK$,X-3,Y-3,X+3,Y+
     3:RETURN
 900 REM CAMBIA IL COLORE DELLO SCHERMO E LO
      PULISCE
 910 GOSUB1020:Q=T
 920 IFQ<1ORQ>8THEN910
 930 COLOR0,Q,I:SCNCLR:RETURN
 940 REM CAMBIA MULTICOLOR 1
 950 GOSUB1020:Z=T
 960 IFZ<1ORZ>8THEN950
 970 COLOR2,Z,I:RETURN
 980 REM CAMBIA MULTICOLOR 2
 990 GOSUB1020:V=T
1000 IFV<1ORV>8THEN990
1010 COLOR3,V,I:RETURN
1020 GETKEY A$:T=VAL(A$):RETURN
```

Funzioni

DEF FN

A questo punto dovreste ormai sapere che una "funzione" è un tipo di comando che prende il valore di una variabile, e opera in qualche modo su di essa prima di ritornarvi un certo risultato. Capita abbastanza spesso di trovarsi a pensare "Non sarebbe comodo se il computer avesse qui una funzione che potesse...". Bene, fortunatamente per noi una tale possibilità esiste, tramite il comando che DEFinisce una FuNzione secondo i desideri dell'utente. Per esempio, potrebbe essere necessario alle volte dover ripetere più volte questo tipo di calcolo:

((ZZ/5)*32)↑2

È possibile allora definire una propria funzione che esegue questo calcolo, così:

10 DEF FNA1(ZZ)=((ZZ/5)*32)↑2

In questo modo si dice al computer di definire una funzione, che avrà il nome FNA1, che divide un certo numero (il valore di ZZ) per 5, lo moltiplica per 32 e eleva il risultato al quadrato. Per vedere come funziona il tutto, provate ad impostare queste linee di programma ed a dare RUN:

```
20 SCNCLR
30 INPUT"BATTI UN NUMERO QUALSIASI";N
40 PRINT"((";N;"/5)*32↑2=";
50 PRINTFNA1(N)
60 GOTO30
```

Come potrete osservare, tutto quello che serve per utilizzare una certa funzione è di fornirle un numero (che va racchiuso fra parentesi), ed il computer provvederà ad eseguire il calcolo usando quel valore. Il computer nel nostro caso sostituirà a ZZ , nel calcolo della funzione, il valore che noi le avremo fornito.

Potete definire un numero qualsiasi di funzioni, purchè abbiano tutte nomi diversi. Il nome di una funzione è formato da FN seguito da una combinazione di lettere e numeri, col vincolo che deve cominciare per una lettera e non deve contenere un comando (proprio come nel caso delle variabili). Per esempio, i nomi FN1A e FNPRINTER non sono accettabili.

Le funzioni che potete definire possono svolgere qualsiasi tipo di calcolo, purchè utilizzino costanti, variabili numeriche e variabili intere. Le variabili stringa o stringhe contenute fra virgolette non possono essere impiegate nelle funzioni definibili dall'utente.

I tasti funzione

Avrete certamente notato sulla tastiera i tasti funzione, marcati da F1 a F7. Questi tasti hanno dei comandi preassegnati, e per sapere di quali si tratta basta che battiate

KEY

ed il computer vi presenterà la lista di quello che i vari tasti funzione svolgono.

Lo stesso comando KEY può però essere adoperato anche per modificare quello che un dato tasto funzione può fare: ossia per ri-definirli. Per esempio, se volete ridefinire il tasto funzione 1 in modo che cancelli lo schermo, e presenti il listato del programma in memoria ogni volta che viene premuto, battete

KEY1,"SCNCLR:LIST"+CHR$(13)

Come vedete, i comandi relativi, SCNCLR e LIST, vanno racchiusi fra virgolette, e se più di uno vanno separati dai due-punti. Il +CHR$(13) alla fine dice al computer che volete l'esecuzione immediata dei comandi quando il tasto viene premuto (CHR$(13) è corrispondente a RETURN, per cui il computer agisce appunto come fosse stato premuto il tasto RETURN alla fine dei comandi in modo diretto).

Può essere necessario a volte inserire qualcosa posto fra virgolette nella definizione dei tasti funzione, e dato che già i comandi sono fra virgolette questo può essere un problema. Per ovviarvi, usiamo il comando CHR$(34) in questo modo:

KEY3,"PRINT"+CHR$(34)+"SALVE"+CHR$(34)+CHR$(13)

Come potete vedere da questo esempio, dove volete inserire le virgolette dovete prima chiudere le virgolette precedenti, aggiungere +CHR$(34), poi aprire le virgolette e continuare col resto.

Potete ridefinire ognuno dei tasti funzione, incluso HELP, per cui se in certi programmi volete che in certi punti con alcuni tasti speciali si possano eseguire delle azioni particolari, potete ridefinire il numero di tasti funzione che vi servono a questo scopo, da uno a tutti.

Funzioni numeriche

La maggior parte delle funzioni numeriche previste dal vostro computer sono già state descritte, ma nei prossimi paragrafi vi esporremo quelle che non abbiamo ancora preso in considerazione.

ABS

La funzione ABS fornisce il valore assoluto, ovvero senza segno di un dato numero. Battete

PRINT ABS(—64)

ed il computer visualizzerà 64 come risultato. Ovviamente un numero positivo rimane inalterato:

PRINT ABS(53)

fornisce ancora 53.

DEC

La funzione DEC converte un valore esadecimale (=in base 16) in numero decimale:

PRINT DEC("F2")

fornisce 242 per risultato.

EXP

La funzione EXP serve per calcolare il valore della costante matematica

“e” (base del sistema di logaritmi naturali, di valore circa 2.71828183...) elevato ad un certo esponente, che ne è l’argomento fra parentesi. Così

PRINT EXP(2)

fornisce il quadrato di e, ossia 7.3890561. Non è una funzione usata spesso, per cui non preocupatevi se di e e di logaritmi naturali non avete mai sentito parlare.

LOG

LOG è l’inversa di EXP, ossia calcola il logaritmo naturale di un certo numero.

PRINT LOG(5)

fornisce il risultato 1.60943791.

SGN

Qualora aveste bisogno di accertare se un certo valore è positivo o negativo, avrete bisogno della funzione SGN. Essa fornisce come risultato -1 se il suo argomento è negativo, 1 se positivo, e 0 se vale 0. Ecco un esempio:

PRINT SGN(-54),SGN(0),SGN(992)

SQR

Questa funzione serve a calcolare la radice quadrata di un numero. Provate con PRINT SQR(169)

ed avrete in risposta 13.

USR

Questa particolare funzione serve a lanciare un programma in linguaggio macchina, ed a passare un numero a tale routine. Per i dettagli sull’impiego di tale funzione, si veda il capitolo dedicato al linguaggio macchina che segue.

Funzioni trigonometriche

Il vostro computer dispone di 4 funzioni trigonometriche fondamentali:

SIN, COS, TAN ed ATN, che, come probabilmente vi aspettate, servono a calcolare il seno, il coseno, la tangente, e l'arcotangente. Se non siete familiari con questi termini, non vi preoccupate - nei vostri programi probabilmente non ne dovrete fare uso; ad ogni modo, il loro significato è spiegato in tutti i testi di matematica per le ultime classi superiori. I comandi grafici che abbiamo già visto coprono la maggior parte delle applicazioni che altrimenti potrebbero richiedere l'uso di queste funzioni.

L'unica cosa importante da ricordare sul modo di operare delle funzioni trigonometriche è che l'argomento delle funzioni SIN, COS e TAN è in radianti. Si tratta di un altro modo per misurare gli angoli: 1 radiante equivale a 57.2957805 gradi (espresso con 7 decimali), ed in un angolo giro (360 gradi) completo ci sono 2*PI radianti. Anche qui, la spiegazione del perchè si usano i radianti al posto dei gradi la potete trovare negli appositi testi di matematica, ma quel che vi serve ricordare per ora è che per convertire un angolo da gradi a radianti si deve dividere per la costante sopra citata: Così se vogliamo calcolare il seno di 40 gradi occorrerà scrivere l'istruzione in questo modo:

```
PRINT SIN(40/57.2957805)
```

Analogamente, se abbiamo il valore della tangente di un certo angolo, e vogliamo sapere qual'è il valore dell'angolo corispondente, la funzione ATN, che serve allo scopo, ci fornisce la risposta in radianti, che dovremo moltiplicare per 57.2957805 per avere il valore dell'angolo in gradi. Se per esempio il valore della tangente è 0.8391, potremo usare

```
PRINT ATN(.8391)*57.2957805
```

che ci fornisce il valore (approssimativamente) di 40 gradi.

Altre funzioni

Ci sono ancora alcune funzioni non numeriche che non abbiamo ancora descritto, cosa che facciamo qui di seguito.

HEX$

Questa è la funzione opposta a DEC, in quanto ci fornisce il valore del numero esadecimale che corrisponde ad un certo numero decimale. Il risultato è una stringa, e come tale può essere assegnato ad una variabile stringa.

```
PRINT HEX$(702)
```

fornisce come risultato 2BE.

FRE

Si tratta di un utile comando, che ci indica quanta memoria rimane ancora libera per i nostri programmi. Se battete

PRINT FRE(0)

vedrete comparire il numero esatto di byte di memoria ancora disponibili. Se lo volete convertire in kilobyte, dividete questo valore per 1024. Il numero fra parentesi non ha significato e può essere qualsiasi: 0 è un valore di comodo.

POS

Questa funzione indica in quale colonna comincerà il prossimo PRINT. Come FRE, fra parentesi può stare un numero qualsiasi, per comodità si usa spesso 0. Ecco un esempio:

Z=POS(0)

SPC

La principale differenza fra SPC e TAB è che SPC opera con la stampante, mentre TAB di solito non può. SPC è perciò impiegato esattamente allo stesso modo di TAB per fare la stessa cosa; cosicchè il comando

PRINT SPC(5);"SALVE"

avrà sulla stampante lo stesso effetto di

PRINT TAB(5);"SALVE"

che visualizza SALVE sullo schermo a partire dalla colonna n.o 5 (la prima è la colonna 0!).

CAPITOLO 11

Il linguaggio macchina

PEEK E POKE

Giunti a questo punto, se avete profittevolmente sperimentato col vostro computer quanto vi è stato insegnato, dovreste avere già una discreta conoscenza del BASIC, e forse non è lontano il momento in cui comincerete a dire: "Se solo potessi muovere la mia astronave un pò più velocemente...", oppure "È possibile effettuare lo 'scroll' anche lateralmente, oltre che verso l'alto o verso il basso?". Vuol dire che è ormai maturo il tempo perchè cominciate ad esplorare il mondo del linguaggio macchina. Il vostro computer dispone di un sistema per rendervi la cosa più facile, che ha nome TEDMON. Prima però di passare ad esaminarlo, bisogna che impariate qualcosa sui modi con cui funziona la memoria del vostro computer, e come possiate determinare i suoi contenuti, o inserirvi delle informazioni.

Il vostro computer dispone di due tipi di memoria: la ROM (Read Only Memory=memoria di sola lettura) e la RAM (Random Access Memory= memoria ad accesso casuale). Il modo con cui queste memorie funzionano è assai complesso, ma cercherò di darvene una spiegazione semplificata.

Con una certa fantasia, provate ad immaginare che dentro il vostro computer ci siano diverse migliaia di scatoline trasparenti, con un numero che le contraddistingue, in modo da poterle individuare (questo numero sarebbe l'indirizzo di una cella di memoria), e che in ognuna sia inserito un foglietto su cui è scritto un numero compreso fra Ø e 255 (questo rappresenta il contenuto di una data cella o locazione di memoria).

Alcune di queste scatoline hanno un coperchio, che non si può aprire; altre invece non hanno coperchio. Le scatoline munite di coperchio rappresentano le celle della ROM, e le altre quelle della RAM.

Dato che le scatoline sono trasparenti, possiamo esaminarne in ogni caso il contenuto, e leggere il numero che sta scritto sul foglietto che sta dentro. Dalle scatoline aperte (la RAM) possiamo anche estrarre il foglietto, ed rimettercene un altro che reca un numero diverso. Con le scatoline chiuse (la ROM) possiamo soltanto leggere il numero che sta all'interno, ma non possiamo in alcun modo modificarlo.

Esistono diversi tipi di ROM. Per prima esiste la CPU o Central Processing Unit (unità di elaborazione centrale), che, come dice il nome, provvede a tutti i generi di elaborazione - è il "cervello" del computer, se così vi piace chiamarlo. Il problema con la CPU è che essa non capisce il BASIC, ma soltanto un proprio linguaggio. Per questo necessita di un altro tipo di ROM, l'"interprete del BASIC", che serve a tradurre le istruzioni BASIC nella forma che la CPU è in grado di comprendere ed eseguire.

Avrete sicuramente notato che quando spegnete il computer ogni programma e dati che esistevano nella sua memoria vanno perduti. Il computer non perde però mai il proprio interprete BASIC: infatti la RAM ha bisogno della tensione di alimentazione per mantenere il proprio contenuto, mentre la ROM è una forma di memoria permanente.

Una parte della RAM viene utilizzata come area di lavoro dal computer. Tutto quanto appare sullo schermo, per esempio, sta in una certa zona della RAM. Pertanto non potete disporre dell'intera RAM per i vostri programmi.

Il comando PEEK (letteralmente: "sbircia") ci consente di esaminare il contenuto di una singola locazione di memoria: ed è possibile usarlo sia per la ROM che per la RAM.

Il comando POKE ("inserisci"), invece, ci permette di cambiare il contenuto di una certa locazione di memoria, ossia di scambiare il foglietto che è contenuto nella relativa scatolina aperta. Non si può eseguire un POKE sulle locazioni della ROM, perchè i contenuti di questa sono inalterabili. Comunque, se tentate di eseguire un POKE su di una cella della ROM, il comando risulterà inefficace, e non si sarà provocato alcun danno.

I comandi PEEK e POKE sono molto utili per modificare lo schermo. Se cancellate quanto c'è sullo schermo, e battete

```
POKE 3072,1
```

vedrete comparire la lettera A nell'angolino superiore sinistro dello schermo. Il numero 3072 è l'indirizzo di memoria di tale quadratino; il numero 1 è il codice ASCII della lettera A. Potete cambiare il 3072 in un altro numero fra 3072 e 4072 e vedrete comparire la A in punti diversi dello schermo. Potete inoltre cambiare il tipo di carattere che viene visualizzato modificando il valore 1 in un altro numero fra 0 e 255.
Se ora provate a battere

```
PRINT PEEK(3072)
```

vedrete comparire in risposta il numero 1 (purchè la A sia sempre sullo schermo). In tal modo abbiamo infatti ordinato al computer di esaminare cosa contiene la cella di memoria 3072 e di visualizzarlo sullo schermo.

Se provate a fare dei PEEK su diverse locazioni comprese fra 0 e 65535 potete controllare cosa contiene la relativa cella (purchè facciate visualizzare il risultato del PEEK). Potete anhe provare a fare svariati POKE, ma non stupitevi se i risultati a volte saranno inattesi.

La memoria del colore

Ora sapete come si possono posizionare dei caratteri sullo schermo tramite dei POKE. È possibile però cambiare pure i colori di ciascun carattere sullo schermo, sempre tramite POKE. Provate a battere

POKE 3072,1:POKE 2048,128

Vedrete comparire la lettera A, lampeggiante, nell'angolo superiore sinistro dello schermo.

Per modificare i colori dei caratteri sullo schermo, e per renderli lampeggianti o togliere il lampeggio, si devono fare tre cose. Primo, dovete decidere il colore del carattere. Il numero relativo sarà un valore fra 1 e 16, come per il comando COLOR. Poi dovete decidere il livello di luminosità, che potrà essere un numero da 0 a 7: questo valore va moltiplicato per 16 e sommato al valore del colore. Se volete pure il lampeggio, dovete aggiungere ancora 128 al risultato. Il valore finale che vi risulta va poi inserito con POKE nella opportuna cella di memoria.

La memoria per il colore va dalla locazione 2048 alla 3048. Il metodo migliore per calcolare dove eseguire il POKE è di sottrarre 1024 dalla locazione di memoria in cui sta il carattere. Se si è fatto un POKE della lettera A nella locazione di memoria 3116, che corrisponde al quadratino posto nella seconda riga dall'alto e nella terza colonna, si deve sottrarre 1024 da 3116, che dà 2092, e quindi fare il POKE del valore di "colore" globale desiderato nella locazione 2092.

Eccovi un programmino che tramite dei POKE visualizza dei circoletti colorati a caso in posizioni a caso dello schermo:

```
10 COLOR4,2,7:COLOR0,2,7:SCNCLR
20 X=INT(RND(0)*999)+1
30 C=INT(RND(0)*15)+1
```

```
40 I=INT(RND(0)*6)
50 C=C+I*16
60 POKEX+2047,C:POKEX+3071,81
70 GOTO20
```

Ecco come funziona questo programma:

Linea 20: scegli un numero a caso fra 0 ed 1, moltiplicalo per 999, ed arrotondalo all'intero inferiore prima di sommare 1; assegna il risultato alla variabile X
Linea 30: scegli un numero a caso fra 0 ed 1, moltiplicalo per 15 e arrotondalo all'intero inferiore, poi somma 1: assegna il risultato alla variabile C
Linea 40: scegli un numero a caso fra 0 ed 1, moltiplicalo per 6 ed arrotondalo all'intero inferiore, poi somma 1: assegna il risultato alla variabile I
Linea 50: moltiplica il valore di I per 16 e somma il risultato a C
Linea 60: memorizza il valore di C nella locazione di memoria 2047+X, poi memorizza il valore 18 nella locazione di memoria 3071+X

Introduzione al TEDMON

Il linguaggio macchina è quello che la CPU interna al vostro computer è in grado di capire. I programmi in linguaggio macchina vengono eseguiti ad una velocità assai superiore a quelli scritti in BASIC, perchè la CPU non ha più bisogno dell'interprete BASIC per tradurre una dopo l'altra le diverse istruzioni.
Questo capitolo non ha la pretesa di essere un'introduzione alla programmazione in linguaggio macchina, perchè un simile argomento richiederebbe un intero libro per suo conto. Vuole invece dare qualche nozione sul TEDMON, un programma "Monitor" scritto in linguaggio macchina che è inserito stabilmente entro il vostro computer, e permette di scrivere programmi in linguaggio macchina. Per richiamarlo basta battere MONITOR. Quello che compare sullo schermo può differire a volte, ma si presenta all'incirca così:

MONITOR (il comando da voi battuto)

MONITOR

```
  PC    SR   AC   XR   YR   SP
;FFFF   00   FF   FF   FF   F9
```

A questo punto TEDMON è diventato operativo, ed i numeri che vedete

sullo schermo, che indicano i vari registri ed i loro contenuti, vi danno certe informazioni sul computer. Vedremo dopo cosa esattamente significano, ma per ora vediamo subito che cosa TEDMON è in grado di fare per noi.

Visualizzazione dei contenuti della memoria

Provate ora a battere

M 8188

Vedrete comparire quel che segue sullo schermo, solo che i caratteri che seguono ai due-punti appariranno invertiti

```
>8188 02 A9 5A 4C 94 04 45 4E:·>ZL..EN
>8190 C4 46 4F D2 4E 45 58 D4:DFORNEXT
>8198 44 51 54 C1 49 4E 50 55:DATAINPU
>81A0 54 A3 49 4E 50 55 D4 44:#
>81A8 49 CD 52 45 51 C4 4C C5:IMREADLE
>81B0 D4 47 4F 54 CF 52 55 CE:TGOTORUN
>81B8 49 C6 52 45 53 54 4F 52:IFRESTOR
>81C0 C5 47 4F 53 55 C2 52 45:EGOSUBRE
>81D0 54 4F D0 4F CE 57 41 49:TOPONWAI
>81D8 D4 4C 4F 41 C4 53 41 56:TLOADSAV
>81E0 C5 56 45 52 49 46 D9 44:EVERIFYD
```

Le lettere che compaiono sull'estrema destra dovrebbero sembrarvi un po' familiari: ne vedremo più avanti il perchè.

All'inizio di ogni linea figura il simbolo/cursore >, che vi consente di modificare gli otto numeri che seguono; anche questa particolarità la vedremo più avanti.

Il numero a 4 cifre che segue immediatamente il > è un indirizzo di memoria, espresso in esadecimale. Come sapete, la nostra normale notazione numerica, quella decimale, prevede 10 diverse cifre per rappresentare i numeri. Nel sistema esadecimale, ovvero in base 16, le cifre diverse sono appunto 16: le cifre da 0 a 9 e le lettere da A ad F, dove A sta per 10, B per 11, e così via sino ad F per 15 (uno meno della base, come 9 è uno meno della base 10). Come abbiamo già accennato, esistono due funzioni del BASIC - HEX$ e DEC - che permettono la conversione da decimale ad esadecimale e viceversa. Se volete quindi convertire questi indirizzi esadecimali, tornate al BASIC ed usate DEC.

Il primo indirizzo, 8188 esadecimale (o 'hex'), corrisponde a 33160 in decimale: si tratta di un indirizzo posto nell'area dell'interprete BASIC. Ciascuna riga comincia come questa con un indirizzo esadecimale, cui seguono 8 numeri esadecimali che indicano i contenuti delle 8 celle successive di memoria che partono dall'indirizzo segnato al principio: così per la prima riga è specificato che la locazione 8188 contiene il numero esadecimale 02, la 8189 A, e così via.

Al termine di ogni riga c'è un gruppo di caratteri invertiti (qui per semplicità rappresentati come normali). Gli 8 numeri esadecimali che li precedono sono i codici ASCII di questi caratteri (se c'è un punto significa che si tratta di un carattere "non stampabile").

La particolare zona di memoria che avete esaminato in questo modo costituisce la tabella dei "termini riservati" dell'interprete BASIC nella ROM (ecco perchè, letti di seguito, i vari caratteri hanno un aspetto familiare: END, FOR, NEXT...) . Se volete seguitare l'esame della ROM battete M seguito da RETURN, e vedrete comparire, listata allo stesso modo, la successiva sezione della memoria. Potete anche impostare

M 8188 8382

che significa che volete esaminare la zona compresa fra gli indirizzi (esadecimali) 8188 e 8382, che costituisce l'intera tabella dei termini riservati. Attenti che lo schermo "scrolla" rapidamente: se volete rallentarlo, tenete premuto il tasto Commodore quando volete fermarlo.

In questo comando, M sta al solito per la chiamata del Monitor; il primo numero esadecimale è l'indirizzo di partenza, ed il secondo quello di fine della zona di memoria di cui volete visualizzare i contenuti (sempre nel formato di righe precedute dall'indirizzo e seguite dai contenuti di 8 locazioni successive).

Abbiamo accennato prima al fatto che il > posto all'inizio di ogni riga vi consente di modificare i contenuti delle locazioni di memoria: basta spostare il cursore dal principio della linea di informazioni, dove si pone automaticamente, sul valore che volete cambiare, e ribattere sopra il nuovo valore. Se provate a fare in questo modo con il "listato" di memoria che abbiamo sopra visto, vedrete che appena premuto il RETURN il numero che avete appena cambiato torna al suo valore originale: questo accade perchè siamo in una zona della ROM, e come sappiamo non è possibile modificarne i contenuti. Se provate invece a battere

M 3000

potrete modificare come vi piace i valori delle varie righe che compariranno (contenuti della memoria a partire dall'indirizzo 3000 hex), perchè stavolta siamo in zona RAM.

Se volete modificare una certa locazione di memoria, non è necessario visualizzarne il contenuto a questo modo, però. Potete usare un comando come questo:

> 324A 3A BD F4

Con esso si dice al computer di memorizzare il numero (hex) 3A nella locazione di indirizzo 324A, il numero BD nella locazione succesiva (324B) e il numero F4 nella locazione seguente (324C).
Con questo sistema potete modificare da 1 ad 8 locazioni di memoria ad un tempo: per es. se volete modificare i contenuti delle 8 locazioni che partono da 2BC2 in avanti, farete

> 2BC2 23 B4 6A DA 9F E2 FC 14

Come uscire da TEDMON

Per abbandonare TEDMON e tornare al BASIC basta battere assai semplicemente

X

ed eccovi rientrati al BASIC.

Riempimento di un'area di memoria

È possibile pure riempire interamente una certa zona di memoria con valori tutti eguali ad uno dato. Per esempio, se si vuole che tutte le locazioni dalla 2400 alla 2A00 contengano il numero hex A3, si batterà

F 2400 2A00 A3

Se ora impostate

M 2400 2A00

potrete verificare come tutte queste locazioni contengano adesso il valore A3.

Ricerca di numeri o stringhe

Un altra comoda particolarità offerta dal TEDMON è la possibilità di ricercare la collocazione in memoria di un certo valore. Ad esempio, se impostate

H 7000 9000 C0

il computer ricercherà nella zona di memoria compresa fra gli indirizzi 7000 e 9000 per ogni presenza del numero C0, e visualizzerà il risultato della sua ricerca, per esempio così:

80AB 83E4 842D 87AB 89A1 8B10 8BDD 8C12 8EB3 8EE1

Ciascuno di questi numeri è un indirizzo di memoria, in cui è contenuto il valore C0 ricercato. Se fate la prova direttamente, vedrete che è proprio così.

Potete anche ricercare la presenza di una certa stringa di caratteri. Se battete

H 8000 9000 'COMMODORE

avrete in risposta

80CF

Se infatti impostate

M 80CF

vedrete che effettivamente la parola "COMMODORE" figura memorizzata in questa zona, con la C di COMMODORE memorizzata nella locazione 80CF.

Nel comando che abbiamo appena utilizzato avrete notato la presenza di un singolo apice (') prima della stringa: esso serve appunto ad indicare al computer che si vuole ricercare una stringa, e non un numero.

Trasferimento di blocchi di memoria

Un altro comando che può risultare assai comodo è quello per il Trasferimento. Se battete

T 0C00 0FFF 0BD8

vedrete che tutto quanto sta sullo schermo si solleva di una riga.

Con questo comando abbiamo in effetti ordinato al computer di trasferire i contenuti della memoria che iniziano dalla locazione 0C00 e finiscono a 0FFF (che costituiscono i limiti della memoria dello schermo) nelle locazioni che partono da 0BD8 in avanti. Dato che 0BD8 si trova 40 locazioni prima di 0C00, questo trasferimento ha per effetto lo 'scroll' verso l'alto di una riga.

Come potete vedere, il comando T di trasferimento va seguito da tre numeri (hex): il primo è l'indirizzo del principio, il secondo quello della fine della zona o blocco da trasferire. Il terzo numero denota il nuovo indirizzo a cominciare dal quale il blocco di memoria va trasferito (spostato).

Scrittura di programmi in linguaggio macchina

TEDMON vi permette anche di scrivere programmi in linguaggio macchina. Provate con la impostazione di questo breve programma in l. m.:

```
A 2000 LDA #$01
A 2002 STA $0C00
A 2005 LDA #$80
A 2007 STA $0800
A 200A BRK
```

Ogni volta che terminate la battitura di una riga, il computer sposterà parte di quanto avete battuto in avanti sullo schermo, ed inserirà alcuni numeri. Automaticamente verranno pure visualizzati una A ed un nuovo indirizzo all'inizio della riga seguente. Una volta terminata la scrittura del programma, segnalatelo battendo il tasto di RETURN. Lo schermo apparirà a questo modo:

```
A 2000    A9 01           LDA>$01
A 2002    8D 00    0C     STA $0C00
A 2005    A9 80           LDA>$80
A 2007    8D 00    08     STA $0800
A 200A    00              BRK
A 200B
```

La lettera A che compare al principio di ogni riga sta per **Assemble**, e dice al computer che quanto scrivete è un'istruzione in linguaggio macchina. Il

numero che segue alla A è l'indirizzo di memoria nel quale va posta l'istruzione successiva: basta indicare la prima locazione, ed il computer sa regolarsi in funzione della lunghezza dell'istruzione.

I numeri che il computer inserisce prima dell'istruzione in linguaggio macchina una volta che premete RETURN sono gli effettivi codici macchina equivalenti ai vari comandi. Ogni istruzione in l. m. si compone di uno o più codici macchina. Per esempio, l'istruzione LDA ha il codice A9 nel programma che stiamo considerando. Il codice cambia a seconda dell'impiego del comando, come vedremo più avanti.

Al codice dell'istruzione seguono uno o due altri codici numerici. Nella prima riga vediamo che LDA #$01 è stato codificato come A9 01: ossia dopo il codice A9 per LDA sta lo stesso valore 01 (in questo caso). Quando si utilizzano gli indirizzi di memoria, come nella seconda istruzione, c'è qualche differenza. Come potete osservare, STA $0C00 diventa 8D 00 0C, il che significa che le componenti 0C e 00 del numero esadecimale dell'indirizzo scambiano posizione rispetto alla solita scrittura: la cosiddetta parte ("byte") "bassa", qui 00, viene posta prima del "byte alto", qui 0C.

Vengono infine i veri e propri comandi. Non intendo qui trattare in dettaglio la programmazione in linguaggio macchina, ma vi darò una breve spiegazione del significato delle diverse istruzioni in l. m. di questo programma:

LDA #$01: carica (LoaD) l'Accumulatore col valore esadecimale 01. Questo accumulatore si può in certa misura assimilare ad una variabile cui viene assegnato un valore

STA $0C00: memorizza (STore) il contenuto dell' Accumulatore all'indirizzo di memoria 0C00 (hex). Questo corrisponde all'angolino superiore sinistro dello schermo

LDA #$80: carica nell'accumulatore il valore esadecimale 80 (128 in decimale)

STA $0800: memorizza il contenuto dell'accumulatore nella locazione di indirizzo 0800. Questa corrisponde alla locazione nella memoria del colore corrispondente all'angolo superiore sinistro dello schermo

BRK: termina il programma e restituisce il controllo al BASIC

Esecuzione di programmi in linguaggio macchina

Il programma ora visto può essere mandato in esecuzione con

G 2000

Non appena premuto RETURN vedrete comparire una A lampeggiante nell'angolo superiore sinistro dello schermo.

Il comando G (GO) ordina al computer di passare all'esecuzione di un programma in linguaggio macchina. Dovete dire al computer da dove parte il programma in memoria, cosa che si fa con il numero posto dopo la G, che costituisce l'indirizzo (qui la locazione 2000 hex) ove è posto l'inizio del programma in l. m. da eseguire.

Disassemblaggio di programmi in linguaggio macchina

Così come vi consente quello che viene detto l'"assemblaggio" di un programma in l. m., ossia la conversione delle varie istruzioni (mnemoniche) in codici macchina, TEDMON può anche permettervi l'operazione inversa, ossia il "disassemblaggio", che converte i codici macchina nelle relative istruzioni. Per questo viene usato il comando D (Disassemblaggio) seguito dall'indirizzo di partenza della zona di memoria che si vuole "leggere". Provate con

D 9000

e vedrete comparire sullo schermo

```
.9000   F0 3C          BEQ   $903E
.9002   C9 FB          CMP   #$FB
.9004   D0 03          BNE   $9009
.9006   4C F7   AE     JMP   $AEF7
.9009   F0 43          BEQ   $904E
.900B   C9 A3          CMP   #$A3
.900D   F0 50          BEQ   $905F
.900F   C9 A6          CMP   #A6
.9011   18             CLC
.9012   F0 4B          BEQ   $905F
.901    C9 2C          CMP   #2C
```

Questo "disassemblato" è nella forma in cui il computer converte il vostro

programma in l. m. quando provvede ad "assemblarlo", come potrete controllare paragonandoli, anche se certi comandi e numeri saranno differenti. Se l'area di memoria che avete sottoposto al disassemblaggio sta nella RAM, potete anche modificarne i contenuti, spostando il cursore sull'istruzione che volete cambiare. Se battete

D 2000 200A

vedrete comparire sullo schermo il vostro programma in l. m. Spostate il cursore sulla prima linea e fate questa modifica:

.2000 A9 01 LDA #$02

Appena premete RETURN questa linea si cambia in

A 2000 A9 02 LDA #$02

Il punto che è posto all'inizio viene trasformato nella lettera A (per Assemblaggio), ed il numero 01 viene modificato in 02. Anche il punto all'inizio della linea successiva si cambia in A, ed il cursore si porta sulla prima cifra del numero (indirizzo) 2002.

Il comando per il disassemblaggio assomiglia a quello per la lettura di una certa zona di memoria, e quindi si scrive

D 4000 4020

per disassemblare ad es. l'area di memoria compresa fra le locazioni 4000 e 4020 (hex).

Confronto fra blocchi di memoria

Un comando forse un pò meno utile è quello per il Confronto fra due blocchi di memoria, che quando viene eseguito segnala dove stanno le eventuali differenze. Per esempio, battendo

C 1000 2000 4000

vedrete lo schermo pressocchè pieno di locazioni di memoria: infatti i contenuti di molte locazioni fra 1000 e 2000 sono diversi da quelli del corrispondente blocco che inizia dalla locazione 4000.

Con quel comando infatti si dice al computer di esaminare il blocco di memoria che inizia dall'indirizzo 1000 hex e termina a 2000 hex, e di

paragonarlo ordinatamente al blocco compreso fra 4000 e 5000 hex (ossia che parte da 4000 ed ha la stessa lunghezza). Nel secondo caso basta indicare l'indirizzo di partenza, perchè il computer sa quanti confronti deve fare.

Il computer visualizza in seguito a questo comando una lista di (indirizzi di) locazioni di memoria i cui contenuti ha trovato diversi nel confronto: per cui solo nel caso in cui i due blocchi di memoria siano identici non visualizzerà alcunchè. Se i blocchi sono alquanto diversi, lo schermo si riempirà invece ben presto di indirizzi.

Salvataggio di programmi in linguaggio macchina

Se avete scritto un programma in linguaggio macchina, vi interessa probabilmente salvarlo su cassetta o su di un disco. Anche a questo TEDMON può provvedere per voi, e basterà che gli segnaliate il nome sotto cui volete salvare il programma, se va memorizzato su cassetta di nastro o su disco, e l'inizio e fine delle locazioni di memoria in cui è attualmente presente. Così, volendo salvare su nastro un programma chiamato LEFT SCROLL (scroll a sinistra), memorizzato fra le locazioni 3000 e 3040, si batterà

```
S"LEFT SCROLL",1,3000,3041
```

Avrete notato l'aggiunta di 1 all'indirizzo di fine del programma in l. m.: ciò dipende dal fatto che la routine di salvataggio memorizza i contenuti di memoria fra l'indirizzo iniziale sino a, ma non compreso, l'indirizzo finale.

Se volete salvare il vostro programma su disco dovete modificare il numero che indica la periferica (l'1 del comando precedente) in 8, così:

```
S"LEFT SCROLL",8,3000,3041
```

Caricamento di programmi in linguaggio macchina

Anche ricaricare un programma in L. M. precedentemente salvato è cosa facile. Dovete solo indicare a TEDMON il nome del programma che volete caricare in memoria, e se lo volete ricaricare da nastro o da disco. Quindi, se volete ricaricare un l.m. salvato col nome FROG (La rana) su nastro, dovete fare

```
L"FROG",1
```

mentre se era stato salvato su disco farete

L"FROG",8

Come in precedenza, l'1 specifica al computer che si tratta di ricaricare un programma da nastro, l'8 che si tratta di un disco.

Verifica di programmi in l. m.

È possibile eseguire la verifica del corretto salvataggio di un programma in l. m. esattamente come era possibile verificare un programma BASIC. Per farlo occorre indicare a TEDMON il nome del programma da verificare, e se esso era stato salvato su nastro o su disco, proprio come nel caso del ricaricamento. Quindi si batterà ad esempio

V"COMPOSER",1

per dire al computer di verificare se il programma salvato su nastro col nome COMPOSER è stato registrato esattamente come figura in memoria. Se i due programmi alla verifica risulteranno identici, riapparirà il cursore lampeggiante; in caso contrario apparirà il messaggio VERIFY ERROR.

Per la verifica di un programma registrato su disco con lo stesso nome, si batterà

V"COMPOSER",8

con gli stessi risultati.

Come nel caso degli equivalenti comandi relativi ai programmi BASIC, le istruzioni di salvataggio, ricaricamento e verifica di programmi in l. m. fanno diventare "bianco" lo schermo se si tratta di nastro, e si ha secondo i casi la comparsa dei soliti messaggi.

I registri

Per finire, parliamo dei registri. Ad essi si riferiscono le lettere ed i numeri che vengono visualizzati quando si richiama TEDMON la prima volta, e che forniscono alcune informazioni relative al computer. Li passeremo ora in esame uno alla volta.

Il primo dei registri ha la sigla PC, per *Program Counter*, ovvero contato-

re di programma. Questo registro "punta" in ogni istante all'indirizzo di memoria dell'istruzione di programma che si sta eseguendo. Per esempio, se il computer sta eseguendo un programma in l. m. che inizia dalla locazione 3A42, il registro PC segna inizialmente 3A42, poi 3A43, quindi 3A44, e via dicendo, a mano a mano che il programma viene elaborato.

Il secondo registro è il registro SR, per *Status Register*, ovvero registro di stato. Questo registro contiene varie informazioni circa le operazioni che sono state appena eseguite (risultati zero, superamento di capacità, ecc.).

Segue il registro *Accumulatore*, che abbiamo già incontrato, e che come già visto si può in certo modo assimilare ad una variabile, anche se si usa in modi diversi da quelle. I registri X ed Y sono simili all'accumulatore, anche se ciascuno ha alcune peculiarità proprie che gli altri non hanno.

L'ultimo dei registri considerati è lo *Stack Pointer*, o puntatore allo stack. Lo stack (il termine viene a volte tradotto con "catasta", ma per lo più rimane in inglese) è un'area di memoria in cui il computer può "salvare" certi numeri, e il suo funzionamento ricorda un pò quello di una pila di libri. Su di questa si possono disporre altri libri, così come se ne possono prelevare: aggiunte e prelievi possono però effettuarsi solo sulla cima della pila di libri, uno alla volta. Il che significa che l'ultimo libro aggiunto in cima sarà il primo ad essere riprelevato, ed in modo del tutto analogo nello stack l'ultimo numero caricato è sempre il primo a venire rimosso. Lo stack ha una capacità massima di 256 byte. Il registro SP punta sempre alla prima locazione libera in cima allo stack.

In ogni istante, potete esaminare il contenuto dei vari registri battendo

R

Verranno allora visualizzati i nomi (sigle) dei vari registri con i relativi contenuti. Se volete, potete modificare il contenuto dei vari registri. Il comando azionato dal tasto del punto-e-virgola serve allo scopo, e dato che esso è già posto all'inizio della riga con i contenuti dei vari registri, basterà spostare il cursore sul contenuto del registro che volete modificare, e ribattere il nuovo valore (in hex). Per esempio, se i registri visualizzati mostrano

```
PC     SR  AC  XR  YR  SP
;BCB1  00  AF  BF  28  F9
```

e volete cambiare il registro X perchè contenga il valore 2A, spostate il

cursore in modo che si sovrapponga alla B di BF, come qui sottto

```
PC     SR  AC  XR  YR  SP
;BCB1  00  AF  BF  28  F9
```

quindi battete 2A e premete RETURN. Il contenuto del
registro X sarà ora 2A.

Il comando SYS

Se volete fare eseguire la vostra routine in linguaggio macchina dal BASIC dovete aggiungere un'istruzione finale RTS in coda. Per esempio, se disassemblate la vostra routine in l. m. e la modificate così

```
A 200A RTS
```

e poi uscite da TEDMON e battete SYS DEC("2000"),vedrete che il vostro l. m. viene eseguito, con la lettera A lampeggiante che compare nell'angolino superiore sinistro.

Il comando SYS, che serve a lanciare un programma in l. m. dal BASIC, deve essere seguito dall'indirizzo di inizio della routine in l. m.: nell'esempio qui sopra abbiamo utilizzato la funzione DEC, che converte il numero 2000 in decimale prima che il comando SYS provochi l'esecuzione del l. m. che parte da tale locazione di memoria.

La funzione USR

Un altro modo per fare eseguire un programma in l. m. a partire dal BASIC è di utilizzare la funzione USR. Questa si può impiegare per passare un numero od una stringa di lettere al linguaggio macchina, e quindi farlo eseguire. Dovete però memorizzare prima l'indirizzo di partenza della routine in l. m. alle locazioni di memoria 1281 e 1282. Per esempio, per eseguire il nostro solito programmino in l. m. tramite il comando USR bisognerà seguire questa procedura:

```
PRINT DEC("2000")
8192

READY
PRINT 8192/256
32
```

```
READY
POKE 1281,0:POKE 1282,32
READY
X=USR(0)
```

ed il programma verrà eseguito.

Quanto abbiamo realizzato in pratica è la conversione del numero esadecimale 2000 in decimale (8192), quindi la divisione di tale numero per 256. Il byte basso (=resto) dell'indirizzo viene inserito con POKE in 1281, ed il byte alto (=quoziente intero), qui 32, in 1282. Dato che USR è una funzione, essa va usata nel formato

variabile = USR(argomento)

Il valore dell'argomento è quello che si vuole passare al programma in l. m. Dato che nel nostro caso non ci necessita di passare alcun valore al programma, il valore in questione non interessa e si può usare 0 per comodità. Quando la routine in l. m. sarà terminata, la variabile di assegnazione di USR conterrà un valore che le viene fornito dalla routine stessa: per cui non usate un nome di variabile che deve essere usata per altri scopi dal BASIC.

A questo punto dovreste aver acquistato un ragionevole grado di conoscenza sugli usi di TEDMON. Se desiderate ampliare le vostre conoscenze sulla programmazione in linguaggio macchina, vi suggerisco di rileggere prima questo capitolo, in modo da essere certi di aver ben capito come impiegare TEDMON per i vostri scopi.

Le unità periferiche

Impiego dei dischi

Le cosiddette "unità disco" sono molto utili perchè vi consentono di salvare e ricaricare i vostri programmi e dati in maniera molto più veloce di quanto è possibile con un registratore a cassette. Potete usare una unità C1541, o una C1542 (l'unica differenza fra le due sta nel colore), oppure l'unità veloce SFS 481, tutte fabbricate dalla Commodore.

Le unità a disco C1541 e C1542 vanno inserite nella presa marcata SERIAL posta sul retro del vostro computer, mentre lo spinotto posto all'altra estremità del cavetto di collegamento va inserito in una delle due prese DIN poste sul retro delle unità disco. Nel caso dell'unità SFS 481 il collegamento va alla presa marcata USER PORT sul retro del computer.

Una volta correttamente collegata l'unità disco, vi servirà naturalmente qualche dischetto su cui salvare i vostri programmi. Il disco adatto è del tipo da 5 1/4" a singola faccia e doppia densità, che può essere settorizzato via hardware o software. Molti rivenditori di computer, ed anche di componenti per l'elettronica e l'informatica, vendono questi dischi.

Precauzioni nell'uso dei dischi

Nell'utilizzo dei dischi occore adottare alcune precauzioni. Il disco in pratica è costuito da un disco di materiale plastico contenuto in un involucro quadrato, concepito per custodirlo all'interno. Nell'involucro è praticata una fessura longitudinale, attraverso la quale potete scorgere il vero e proprio disco. Evitate accuratamente di toccarne la superficie che compare attraverso la fessura. Il disco è rivestito di materiale magnetico simile a quello che ricopre i nastri delle cassette audio, e, come queste, un disco non va mai posto in vicinanza ad una calamita o fonte di magnetismo (il che include il vostro televisore o monitor, e la parte superiore dell'unità disco stessa), nè esposto a fonti di calore o alla luce solare diretta. I dischi vanno conservati ad una temperatura fra 10 e 50 gradi centigradi, e si deve evitare di piegarli in alcun modo.

Protezione contro la scrittura

Prima di poter salvare alcunchè sul disco dovete procedere a "formattarlo". Prima di tutto dovete accertarvi che la piccola tacca quadrata che vedete sulla sinistra dell'involucro non appaia coperta. Questa tacca ha una funzione simile alla linguetta presente sulle cassette di nastro, che impedisce la registrazione sulle cassette da cui sia stata asportata sul retro: analogamente se questa tacca è stata ricoperta, sul disco non si può salvare niente, nè modificarne in alcun modo i contenuti. Ecco perchè essa ha ricevuto il nome di "Write protection tab", ossia tacca di protezione contro la "scrittura" sul disco.

Una volta sicuri che la tacca è libera, potete introdurre il disco nell'unità disco. Aprite il portellino di questa, premendolo verso l'interno e sollevandolo. Inserite quindi il disco con la tacca di protezione sulla sinistra, e la fessura attraverso cui si scorge il disco rivolta verso il fondo. Quindi chiudete lo sportello abbassandolo: se l'operazione è corretta lo sentirete incastrarsi con un 'click'.

Inizializzazione di un disco

Ora che avete inserito correttamente il disco entro la sua unità potete passare alla sua inizializzazione. Il computer si aspetta che il disco venga suddiviso in tracce e settori: ci sono 35 tracce con da 17 a 21 settori per ogni traccia. Ogni settore può contenere 256 byte. Per 'formattarè in questo modo il disco si deve usare il comando HEADER. Quindi, una volta certi di avere un disco entro l'unità disco, battete

HEADER "DISK1",I01,D0,U8

Il computer vi porrà la domanda

ARE YOU SURE? ("Sei proprio sicuro?")

a cui risponderete normalmente Y per "yes" (SI), perchè volete effettivamente inizializzare il disco in questo caso. Vedrete allora che il motore del disco si avvia, e accendersi la spia rossa. Ci vuole un discreto tempo per inizializzare un disco, quindi munitevi di pazienza ed attendete sino a quando il disco si ferma.

Se la spia rossa continua a lampeggiare mentre il disco ha finito di girare,

significa che il disco non ha ricevuto una corretta inizializzazione. Per trovare cosa è andato storto, battete

PRINT DS$

Il computer vi fornirà un messaggio di errore, e la spia rossa smetterà di lampeggiare. Dovete allora controllare se il disco era stato inserito propriamente, e se la tacca anti-scrittura era libera. Provate a ripetere la procedura di inizializzazione un altro paio di volte, e se ancora non dovesse funzionare provate con un altro disco. Se questo secondo disco si inizializza correttamente senza intralci, significa che il primo era in qualche modo avariato. Se dopo diversi tentativi con dischi differenti, non riuscite in alcun caso ad inizializzare un disco, vuol dire che è responsabile l'unità disco, che sarà bene riportare al rivenditore per la sostituzione.

Quello che avete in sostanza ordinato di fare al computer battendo il comando sopra citato è di suddividere il disco nel giusto numero di tracce e di settori, e di assegnare al disco il nome DISK1 (o naturalmente un altro nome a vostra scelta); inoltre di dare al disco il numero di identificazione 01, che il computer ha provveduto a collocare entro ciascun settore. Il D0 dice al computer che l'inizializzazione deve avvenire nell'unità disco 0 (la prima), e l'U8 finale corrisponde al fatto che le unità disco corrispondono al numero di periferica 8.

Il catalogo (DIRECTORY) del disco

Oltre a suddividere il disco nel giusto formato, il comando HEADER serve anche a creare un indice, o catalogo (DIRECTORY in inglese) sul disco. Potete determinare quali sono i programmi incisi su di un dato disco impostando

DIRECTORY

Dato che per ora il nostro disco è ancora vuoto riceverete in risposta sullo schermo

0"DISK1 "01 2A
664 BLOCKS FREE

DISK1 è il nome assegnato al disco, e 01 il numero di identificazione ad esso attribuito. Il 2A alla fine della prima riga è il numero di identificazione attribuito dal computer al disco. Vedete inoltre segnalato il numero di settori liberi del disco.

Potete anche farvi mostrare tutti i nomi dei file presenti che iniziano con una certa lettera o gruppo di caratteri. Per esempio, se volete che vi siano mostrati tutti i file su di un disco il cui nome comincia per PROG battete

DIRECTORY "PROG*"

con il gruppo PROG racchiuso fra virgolette, e seguito da un asterisco: questo asterisco dice al computer che volete i nomi dei file presenti sul disco che cominciano per PROG.

Potete rallentare la velocità con cui viene visualizzato il catalogo del disco, per una più comoda lettura, premendo il tasto Commodore. Premendo CONTROL ed S la lista si ferma, e può esser fatta ripartire premendo un altro tasto qualsiasi.

Salvataggio di un programma su disco

Ed ora siete pronti per il salvataggio di un programma sul disco. La cosa più semplice a questo punto è di caricare uno dei vostri programmi registrati su cassetta, ed una volta fatto questo impostare sulla tastiera

DSAVE"Nome programma"

ovviamente con il vero nome che volete per il programma. Esso può essere costituito da un qualsiasi gruppo di lettere e numeri, purchè inizi per una lettera e non superi i 16 caratteri totali. Il comando DSAVE svolge le stesse funzioni di SAVE, tranne che salva il programma sul disco invece che su nastro: inoltre lo schermo non diventa "bianco" quando si salva su disco.

Verifica di un programma

Una volta che vedrete comparire READY, potete verificare che il programma è stato salvato correttamente battendo

VERIFY"Nome programma",8

dove al solito sta il vero nome del programma; e l'8 finale si riferisce all'unità disco come periferica. Con VERIFY, come nel caso del nastro, si dice al computer di verificare la corrispondenza fra quanto è stato memo-

rizzato sul disco e quanto è ancora presente in memoria, con riferimento al programma di dato "nome". Il VERIFY funziona esattamente come per il nastro, ma anche qui lo schermo non viene "sbiancato" per la durata dell'operazione.

Se volete, potete battere DIRECTORY e controllare il catalogo dei nomi dei file registrati sul disco, per vedere come il nome del nuovo programma sia stato aggiunto alla lista. Il numero che trovate davanti al nome del programma è il numero di blocchi (settori) che il programma ha "consumato" sul disco (ogni blocco è lungo 256 byte, per cui dal prodotto "256 x n.o settori" potete avere una stima della lunghezza del programma in byte).

Caricamento da disco

Ora che siete sicuri che il vostro programma è stato salvato sul disco, potete sperimentarne il ricaricamento. Battete NEW per cancellare il programma dalla memoria del computer e poi

DLOAD"Nome programma"

In un tempo sorprendentemente breve il vostro programma verrà ricaricato, pronto per il RUN, LIST o quanto altro volete fare con esso.

Non è strettamente necessario ribattere l'intero nome del programma quando volete ricaricarlo. Per esempio, se battete

DLOAD"RANA*"

il computer caricherà il primo programma che incontra e comincia per RANA. In alternativa, se sapete che il programma che vi serve è il primo del disco, potete battere

DLOAD"*"

Modifica del nome di un programma

È assai facile, ed a volte utile, cambiare il nome che è stato assegnato ad un certo programma nel salvataggio su disco. Basta usare il comando RENAME, specificando al computer che si vuole assegnare ad un certo programma di dato nome un nome diverso. Per esempio, se avevate sul

disco un programma chiamato INVADER e volete modificarne il nome in ALIENI, basterà battere

RENAME"INVADER"TO"ALIENI"

Potete ridenominare qualsiasi file del disco con questo comando RENAME, e la cosa richiede solo pochi secondi, in quanto il computer deve solo procedere ad aggiornare il catalogo.

Come fare una copia extra di un programma

Spesso è utile disporre di una copia in più di un programma particolarmente importante, in modo da avere una copia di riserva nel caso venisse accidentalmente cancellata o danneggiata la prima registrazione. Una maniera per ottenere tale scopo è di usare il comando COPY. Supponendo ad esempio di volere una seconda copia di un programma chiamato OROLOGIO, batterete

COPY"OROLOGIO"TO"OROLOGIO2"

Come vedete, il nome assegnato alla seconda copia differisce lievemente da quello originale: infatti non è possibile avere due file sullo stesso disco con il medesimo nome. COPY richiede un pò di tempo, tanto più quanto più lungo il programma o file.

Se disponete di più di un'unità disco collegata, potete anche ricopiare con COPY un programma da un disco ad un altro. Se volete p. es. avere una copia di OROLOGIO, posto sul disco nell'unità 0, in quello presente nell'unità 1, si batterà

COPY D0,"OROLOGIO"TO D1,"OROLOGIO"

In questo caso è possibile dare alla seconda copia lo stesso nome di file, perchè i due stanno su dischi diversi.

Il comando COPY può anche essere usato per ricopiare ogni cosa che sta su di un disco su di un secondo, se sono collegate due unità disco. Basta fare

COPY D0 TO D1

Cancellazione di un programma da un disco

In certi casi può essere necessario eliminare un certo programma da un disco: a tal fine si usa il comando SCRATCH. Per esempio, se volete cancellare un programma di nome AVVENTURA, battete

SCRATCH"AVVENTURA"

Il computer - per sicurezza sulle vostre reali intenzioni - vi chiederà

ARE YOU SURE?

a cui risponderete Y (o N, se avete nel frattempo cambiato idea, per NO...).

Ri-salvataggio di un programma

Quando volete salvare una versione aggiornata di un programma al posto di quella originale, dovete aggiungere una @ prima del nome del programma nel comando DSAVE. Per esempio, se volete sostituire alla versione del programma RANA attualmente sul disco una sua versione modificata, battete

DSAVE"@RANA"

ed il computer cancellerà dal disco la precedente versione di RANA e salverà al suo posto la nuova.

Compattazione di un disco

Se avete avuto occasione di usare più volte uno stesso disco, salvando e cancellando diversi programmi, su questo si saranno gradatamente prodotte certe lacune. Inoltre, se avrete avuto a che fare anche con la gestione di file (di cui parleremo più avanti), ci possono essere vari file mal chiusi dispersi lungo il disco. Per dare un certo "riordino" a tutto questo materiale, e "compattare" assieme i programmi in modo da occupare il minor spazio possibile sul disco, conviene usare di tanto in tanto il comando

COLLECT

che non richiede nient'altro - se non la pressione di RETURN dopo averlo impostato. Il disco si metterà in moto e, dopo un certo tempo, risulterà rimesso in ordine, assicurandovi maggior spazio per salvare i futuri programmi.

Ottenimento di copie di BACKUP

A questo particolare scopo esiste il comando di BACKUP, utilizzabile quando si dispone di almeno due unità disco separate: questo comando copia il contenuto di un intero disco su di un altro. Per esempio, per preparare una copia di riserva di quanto si trova sul disco dell' unità 0 su di un disco posto nell'unità 1, si batterà

BACKUP D0 TO D1

Il computer procederà a ricopiare il contenuto del primo disco sul secondo, cancellando quanto eventualmente fosse memorizzato su quest' ultimo. Il secondo disco non richiede neppure di essere stato inizializzato, perchè anche a questo provvede il comando BACKUP.

Tutti i comandi relativi ai dischi si possono usare in egual modo sia che si disponga di una sola, o due, o più di due unità disco ("drive" come talvolta vengono detti). Per utilizzare correttamente i comandi, tranne che nel caso dell'unità 0, è necessario specificare ogni volta con D seguito dal numero dell'unità a quale "drive" ci si riferisce: questa specificazione va posta alla fine del comando. Per esempio, se si vuole il "catalogo" del contenuto del disco presente nell'unità 1 (secondo disco) si farà

DIRECTORY D1

e per caricare in memoria un programma dal terzo "drive" si batterà

DLOAD"nome programma", D2

Uso di una stampante

Una stampante costituisce un dispositivo assai utile per ottenere copie permanenti su carta (la cosiddetta "hard copy") dei listati di un programma, che possono essere conservate in un archivio per il caso di perdita del programma originale.

La maggior parte delle stampanti idonee all'impiego con il vostro computer sono costruite dalla Commodore (state attenti con le altre, perchè alcune non sono in grado di stampare i simboli grafici, ed altre abbisognano di interfacce e cavi speciali). La stampante va collegata alla presa marcata SERIAL posta sul retro del computer. Per i dettagli relativi ad un corretto collegamento della stampante alla rete ed al computer, si vedano le istruzioni del relativo manuale d'uso.

Prima di poter inviare qualsiasi cosa alla stampante si deve aprire (OPEN) un canale tramite il quale il computer possa trasmettere le informazioni alla stampante. Per fare questo, si deve usare un comando del tipo

OPEN 1,4

che significa "aprire un canale con il dispositivo periferico n.o 4, la stampante, ed assegnargli il numero di file 1". Ogni volta che si vuole inviare qualcosa alla stampante dovete riferirvi ad essa col numero di file prescelto. Il numero di file può essere un valore fra 1 e 255, per cui il comando poteva anche essere della forma

OPEN 54,4

Il motivo per cui si usa un numero di file è che in uno stesso istante possono risultare aperti più di un canale contemporaneamente, e l'assegnazione ad ogni canale aperto di un numero diverso rende più facile passare dall'uno all'altro di essi, come vedremo.

Al termine del comando OPEN può stare anche un terzo numero: esso costituisce l'*indirizzo secondario*, e può andare da 0 a 7 quando si usa la stampante. La scelta di un indirizzo secondario pari a 0 dice al computer che volete la stampa in caratteri maiuscoli, mentre l'indirizzo secondario 7 significa stampa in lettere minuscole.

Una volta aperto con OPEN un canale, dovete decidere che cosa inviare alla stampante. Se volete stampare solo un limitato numero di messaggi, potete usare il comando #: per ora supponiamo sia questo il caso. Provate allora con questo esempio:

OPEN 1,4:PRINT#1,"PROVA DI STAMPA"

Non appena premuto il tasto RETURN la stampante procederà a stampare sulla carta PROVA DI STAMPA. Se il canale con il numero di file 1 era già stato aperto in precedenza (e non più chiuso), riceverete un messaggio di errore: in questo caso battete prima CLOSE 1 e ripetete.

Avrete notato che abbiamo impiegato il numero di file usato nell'aprire il canale anche nel comando PRINT#: infatti ci si deve sempre riferire ala stampante con il suo numero di file, e non con il numero di unità.

Sulla stampante è possibile stampare qualsiasi cosa veniva normalmente visualizzata ("stampata") sullo schermo, però vi facciamo notare che

PRINT USING e PRINT TAB non funzionano come con lo schermo. Di questo parleremo più oltre.

Una volta finito di stampare i vostri messaggi, dovete chiudere (CLOSE) il canale utilizzato, così:

CLOSE 1

che procede a chiudere il canale col numero di file 1, tramite il quale non si può quindi più inviare informazioni alla stampante.

Il secondo modo con cui si possono trasmettere informazioni alla stampante è tramite il comando CMD. Esso serve per indicare al computer di inviare alla stampante tutto quello che normalmente andrebbe visualizzato sullo schermo. Per cui se battete un programmino e poi fate

OPEN 1,4:CMD:LIST

il computer aprirà un canale con numero di file 1 per la stampante, e poi listeràsullo schermo il programma. D'ora in avanti però qualsiasi cosa sarebbe finita sullo schermo apparirà invece sulla carta della stampante. Così, se battete

PRINT"QUESTA È UNA SECONDA PROVA DI STAMPA"

vedrete comparire QUESTA È UNA SECONDA PROVA DI STAMPA sulla carta.

Una volta listato il programma, dovete procedere ancora a chiudere il canale, ma prima di fare ciò dovete dire al computer di non inviare più informazioni alla stampante, e di rinviarle allo schermo. Per questo battete

PRINT#1

comando che farà visualizzare nuovamente sullo schermo i vari testi: tuttavia il canale è rimasto ancora aperto. Per richiuderlo, dovete fare

CLOSE1

Gestione di file su cassetta

Spesso risulta utile poter memorizzare dati su nastro, specie quando si sta scrivendo un lungo programma e ci si accorge che la memoria scarseggia.

Fortunatamente, come sappiamo, è possibile salvare informazioni su nastro, e ricaricarle assegnandole a delle variabili.

Prima di salvare qualsiasi informazione su nastro dobbiamo prima aprire (OPEN) un canale per il nastro, nello stesso modo con cui si apre per un'operazione su disco: per cui battete

```
10SCNCLR
20OPEN 3,1,2,"FILE1"
```

Il comando OPEN di linea 20 dice al computer di aprire un canale riservato al nastro (unità 1) e di assegnarli il numero di file 3. Il secondo dei tre numeri è appunto il numero della periferica, che per il registratore a nastro è 1, ed il terzo un numero che specifica al computer che cosa intendete fare con quel file. Il significato di tale numero è

0 Input (ingresso dati)
1 Output (uscita dati)
2 Output completato con un marcatore di "fine nastro"
(End Of Tape=EOT)

Nel nostro caso abbiamo optato per la selezione di "output con marcatore EOT" (2). Questo segnale indica al computer che quando ha terminato di memorizzare informazioni sul nastro deve incidere un "marcatore" per indicare che sullo stesso non ci sono più altre informazioni. Se si trascura l'aggiunta di questo marcatore, il computer seguiterebbe a ricercare informazioni, anche dopo aver raggiunto la fine di un file.

Ora che abbiamo aperto il nostro file, dobbiamo trasmettergli qualcosa. Per questo si impiega anche per il nastro il comando PRINT#, in modo simile a quello che si è fatto per trasmettere dati ad una stampante. Aggiungete perciò queste linee alle precedenti:

```
30 PRINT#3,"SALVE";CHR$(13);"A TUTTI";CHR$
   (13);"QUANTI"
40 PRINT#3,1;CHR$(13);2;CHR$(13);3
50 CLOSE3
```

Con la linea 30 diciamo al computer di trasmettere le stringhe SALVE, A TUTTI, QUANTI al file numero 3, ossia al file che abbiamo aperto per il registratore a nastro. Ogni stringa è separata mediante un CHR$(13), ossia l'equivalente della pressione di RETURN. La sua funzione è appun-

to di separare i vari termini (dati) sul nastro: il computer necessita di questo carattere di controllo come separatore per evitare ogni confusione all'atto del ricaricamento in memoria.

La linea 40 assomiglia alla 30, salvo che al nastro vengono inviati i dati numerici 1,2 e 3 invece di stringhe. Anche i numeri vanno separati con dei CHR$(13).

Una volta terminato l'invio delle informazioni al nastro, dobbiamo chiudere il canale, cosa affidata alla linea 50. Il comando CLOSE è anch'esso usato esattamente nello stesso modo di quando si chiudeva un canale per la stampante.

Ed ora fate girare il programma. Prima però inserite un nastro nel registratore, e riavvolgetelo sino a poco prima dell'inizio della parte magnetica (qualora ci fosse il tratto "leader" non magnetizzabile a ciascun estremo). Date il RUN, e fate partire il nastro in registrazione premendo i tasti PLAY e RECORD assieme, secondo il solito. Lo schermo diventerà "bianco" mentre il computer procede a memorizzare le informazioni sul nastro.

Una volta registrati i dati, battete NEW per cancellare la memoria, e ribattete questo programma:

```
10 SCNCLR
20 OPEN3,1,0,"FILE1"
30 INPUT#3,A$,B$,C$
40 INPUT#3,A,B,C
50 CLOSE3
60 PRINTA$;"  ";B$;"  ";C$
70 PRINTA;B;C
```

Con la linea 0 si apre un canale di Input per il registratore a nastro: il primo numero del comando OPEN è il numero del file, poi viene il numero della periferica (1), e quindi per terzo un numero che indica al computer che in questo caso si vogliono ricevere informazioni (Input) da tale unità periferica. FILE1 è il nome del file.

Con le linee 30 e 40 si "leggono" le informazioni dal nastro usando il comando INPUT#. Questa istruzione dice al computer di leggere informazioni da una periferica diversa dalla tastiera, e per il resto opera in modo simile al o si dice al computer di leggere tre stringhe, e di assegnarle alle variabili stringa A$, B$ e C$, mentre nella linea 40 si fanno leggere tre numeri, da assegnare alle variabili A, B e C.

Poi viene utilizzato il comando CLOSE per richiudere il canale; infine le linee 60 e 70 visualizzano sullo schermo le informazioni così raccolte.

Ora riavvolgete ancora una volta il nastro e date il RUN: il computer vi segnalerà PRESS PLAY ON TAPE ("Premi il tasto PLAY per l'ascolto") e fatto questo lo schermo si cancellerà sin tanto che il computer legge le informazioni sul nastro. Poi lo schermo torna normale, e verranno visualizzate le informazioni lette dal nastro.

Un altro metodo per ritrovare le informazioni registrate su di un nastro è tramite l'istruzione GET#: essa è simile ad INPUT#, ma legge un carattere per volta. Provate a battere questo programma:

```
10 SCNCLR
20 OPEN3,1,0,"FILE1"
30 FORN=1TO25
40 GET#3,A$
50 PRINTA$;:NEXT
60 CLOSE3
```

Quando la farete girare vi verrà ancora richiesto di mettere in moto il nastro in PLAY, ma questa volta il computer leggerà le informazioni sul nastro un carattere alla volta, ed allo stesso modo le visualizzerà sullo schermo.

Il comando GET# di linea 40 legge le informazioni sul nastro variabile A$. Essa viene poi visualizzata ogni volta.

Dato che GET# legge un carattere alla volta, è capace di leggere anche CHR$(13), per cui quando dovrebbe visualizzare questo particolare carattere agirà come se aveste premuto il tasto di RETURN.

Sommario

Il comando OPEN serve per aprire un determinato canale ad un'unità periferica, tramite il quale la stessa possa ricevere od inviare informazioni.

L'istruzione PRINT# serve per trasmettere dati ad una periferica tramite un canale previamente aperto con OPEN.

Le informazioni inviate da una periferica si possono leggere mediante l'istruzione INPUT#, che legge tutto un gruppo di caratteri insieme.

Un metodo alternativo di lettura è tramite l'istruzione GET# che però legge solo un carattere per volta.

Una volta terminato di utilizzare una periferica bisogna chiudere (CLOSE) il canale attraverso il quale sono avvenute le comunicazioni con essa.

Gestione di file su disco

Benchè la memorizzazione delle informazioni su nastro possa risultare utile, essa ha il difetto di essere lenta. I dischi risultano invece molto più veloci, e quindi sono l'ideale per la gestione dei file.

Il salvataggio delle informazioni su disco segue una procedura molto simile a quella vista per il nastro, con le opportune modifiche circa il numero della periferica. Provate ad impostare questo programma:

```
10 SCNCLR
20 OPEN1,8,2,"FILE SU DISCO,S,W"
30 PRINT#1,"QUESTO E' UN BREVE MESSAGGIO";
   CHR$(13);"E QUESTO INVECE UN ALTRO."
40 PRINT#1,1;CHR$(13);2;CHR$(13);3
50 CLOSE1
```

Il comando OPEN viene usato per il disco in modo del tutto simile che per la cassetta: il primo numero posto dopo OPEN è il numero di file, il secondo il numero della periferica (solitamente 8 per l'unità disco), ed il terzo il numero del canale per i dati, che può essere un valore fra 2 e 14. Le lettere poste fra virgolette costituiscono - tranne la parte 'S,W' - il nome del file. La S dice al computer che si vuole impiegare un file di tipo sequenziale, ossia un file in cui tutte le informazioni vengono memorizzate nell'ordine in cui sono introdotte. La W sta per Write, ovvero "scrittura", a denotare che si tratta di un'operazione di registrazione di informazioni da inserire nel dato file.

Le informazioni vengono inviate al disco esattamente come nel caso del nastro. Le linee 30 e 40 istruiscono il computer di trasmettere le informazioni con il comando PRINT#: anche qui i vari "pezzi" di informazione vanno separati tramite CHR$(13).

Naturalmente si deve chiudere (CLOSE) il file una volta terminato l'uso, cosa che avviene ad opera della linea 50.

Se date il RUN al programma, il disco si metterà in moto automaticamente, per registrare le informazioni. Data la velocità del disco, questa operazione prende poco tempo.

La rilettura delle informazioni dal disco è altrettanto semplice. Fate NEW e impostate il programma:

```
10 OPEN1,8,2,"FILE SU DISCO,S,R"
20 INPUT#1,A$,B$,A,B,C
30 CLOSE1
40 PRINTA$:PRINTB$:PRINTA;B;C
```

Con la linea 10 si apre un canale per il disco come nel primo programma, con l'unica differenza che dopo il nome del file troviamo i caratteri 'S, R' ad indicare che si vuole leggere (Read) da un file Sequenziale.

Le informazioni vengono lette in linea 20 tramite l'istruzione INPUT#, allo stesso modo che con il nastro. Poi il file viene chiuso in linea 30, prima che la linea 40 proceda a visualizzare le informazioni lette.

Come nel caso dei file su nastro, si può usare anche GET# per la lettura dei dati da un disco. Battete il programma che segue:

```
10 OPEN1,8,2,"FILE SU DISCO,S,R"
20 FORN=1TO59
30 GET#1,A$:PRINTA$;
40 NEXT:CLOSE1
```

Il comando GET# agisce esattamente come nel caso del nastro, leggendo da un file su disco un carattere alla volta, come potrete controllare dando il RUN a questo programma.

Sommario

Quando si apre con OPEN un file dati su disco bisogna specificare che si desidera una file sequenziale (S), e se l'operazione da eseguire sul file è di lettura (R) o scrittura (W).

Per leggere "in un sol colpo" informazioni relativamente lunghe si usa l'istruzione INPUT#, come nel caso del nastro.

Invece, anche qui come per il nastro, per leggere le informazioni dal disco un carattere alla volta si usa GET#

Bisogna sempre chiudere con CLOSE il canale che si è aperto all'inizio delle operazioni su di un file al termine delle stesse.

Lista dei termini BASIC

Vi forniamo un elenco di tutti i comandi, istruzioni e funzioni che avete imparato a conoscere, accompagnati da una breve descrizione sul funzionamento. Quando si trova una lettera minuscola, come in ABS(a), significa che essa può essere sostituita da un valore numerico. Se una parte di comando figura racchiusa fra parentesi quadre, come in CIRCLE s,xr [,yr,sa,ea,ra,de] ciò sta ad indicare che tale parte è opzionale.

ABS(a) - Converte il dato numero in valore assoluto (positivo)

AND - Si può impiegare assieme ad IF...THEN, per es. IF A=1 AND solo se il valore di A è 1 e contemporaneamente quello di B è eguale ad 1. AND , che visualizzerà la lettera A è pure un operatore Booleano, per es PRINT 3 AND 2 visualizza 2.

ASC("a") - Ritorna il codice ASCII di un dato carattere

ATN(a) - Ritorna il valore dell'arcotangente dell'angolo a

AUTO(n) - Attiva l'auto-numerazione delle linee. Il valore di n indica l'incremento fisso fra i numeri di linea. Se si imposta AUTO non seguito da alcun valore si disattiva la numerazione automatica

BACKUP Dn TO Dm [,ON Uz] - Copia il contenuto del disco collocato nell'unità disco n sul disco posto nell'unità m. Si può anche specificare il numero di drive.

BOX cs,x1,y1,x2,y2 - Disegna un rettangolo sullo schermo nel colore sorgente cs, con un vertice alle coordinate x1,y1 ed il vertice opposto alle coordinate x2,y2

CHAR cs,x,y,"stringa" - Visualizza la stringa di caratteri posizionata col suo primo carattere nelle coordinate x,y, nel colore sorgente cs

CHR$(x) - Ritorna il carattere che ha il codice x

CIRCLE cs,x,y,xr [,yr,sa,ea,ra,de] - Disegna un circolo od un'ellisse, nel colore sorgente cs, con il centro alle coordinate x,y, ed il raggio xr. Se

è indicato yr, si tratta del semiasse verticale: in sua assenza si assume il valore eguale ad xr. Si può tracciare solo un arco di cerchio specificando l'angolo di origine sa e di termine ea. La figura può essere rotata di un angolo ra, e altre figure (poligoni) si possono tracciare specificando l'angolo de fra ciascun segmento.

CLOSE f - Chiude il file numero f

CLR - Reinizializza i valori di tutte le variabili

CMD f - Invia tutte le informazioni al file numero f invece che allo schermo

COLLECT [Dn, ON Uz] - Compatta tutti i file di un disco, e cancella i file non correttamente chiusi con CLOSE. Possono essere specificati il numero del drive e della periferica

COLOR cs,cl,lu - Assegna il colore cl ed il livello di luminosità lu al colore sorgente cs

CONT - Fa ripartire l'esecuzione di un programma che si era fermato per effetto di istruzioni STOP od END, o se era stato premuto il tasto RUN/STOP. Se è intervenuto un errore oppure una linea è stata modificata in qualche modo, CONT non ha effetto

COPY [Dn,] "file 1" to [Dm,] "file 2" [,ON Uz] - Esegue una copia di un programma, sullo stesso disco o su di un altro disco posto in un altro drive

COS(n) - Ritorna il coseno dell'angolo n

DATA lista dati - Serve per avere disponibile una lista di dati entro il programma, che può essere letta quando conviene

DEC("hn") - Converte in numero decimale la stringa esadecimale hn

DEF FNva - Definisce una funzione di nome FNva (va è un nome lecito per una variabile)

DELETE [n. linea iniz.][,n. linea finale] - Serve per cancellare un blocco di linee di istruzioni

DIM lista variabili - Riserva lo spazio di memoria necessario per una o più variabili multiple

DIRECTORY [Dn,Uz,"nome file"] - Visualizza i nomi dei file registrati sul disco posto nel drive n sull'unità disco z. Se è indicato il "nome file" verranno visualizzati tutti i file il cui nome inizia con quei caratteri

DLOAD"nome programma" - Carica da disco un programma di nome specificato

DO [UNTIL arg. Booleano/WHILE arg. Booleano] istruzioni di prog. [EXIT] LOOP [UNTIL] arg. Booleano/WHILE arg. Booleano] - Procede all'esecuzione di tutte le istruzioni del programma poste fra DO e LOOP sino a quando (UNTIL) o fintanto che (WHILE) una certa condizione è verificata. La condizione va espressa come argomento Booleano

DRAW [cs[,x1,y1] TO x2,y2]/[cs[,x1,y1] TO d;a] - Traccia una linea (segmento) dal punto x1,y1 sino al punto x2,y2, nel colore sorgente cs. In alternativa questo comando si può impiegare per tracciare un segmento lungo d pixel a partire dal punto x1,y1, sotto un angolo di a gradi con la verticale

DSAVE"nome programma" - Salva il programma in memoria su di un disco con un nome specificato

ELSE istruzioni - Usato con le strutture IF...THEN (vedi IF)

END - Istruisce il computer di arrestare il programma

ERR$(en) - Visualizza il messagio di errore corrispondente al codice di errore en

EXIT - Serve per uscire anzitempo da un DO...LOOP

EXP(x) - Ritorna il valore di e elevato ad x

FRE(x) - Ritorna il numero di byte ancora liberi in memoria per il programma BASIC. x può essere un valore qualsiasi (p. es. 0)

FOR n=st TO en [STEP in]:istruzioni:NEXT n - Esegue le istruzioni comprese fra FOR e NEXT sino a che la variabile n, che parte dal valore st e si incrementa di in ad ogni passaggio, non raggiunge o supera il valore en. Se manca STEP l'incremento è fissato automaticamente ad 1.

GET[KEY] var$ [,var$,var$,...] - Sonda la tastiera e memorizza nella variabile var$ il carattere del tasto che fosse premuto. Se c'è pure KEY

(GETKEY var$), il computer ferma l'esecuzione del programma sino a quando non viene premuto un tasto. Sia GET che GETKEY si possono impiegare per la pressione in successione di più tasti, facendoli seguire da una lista di variabili stringa di assegnazione

GRAPHIC modo [,clear]/CLR - Serve per selezionare uno dei modi grafici. Il modo va scelto fra 0 e 4, e clear deve essere 0 (per lasciare lo schermo come sta) oppure 1 (per cancellare lo schermo). L'istruzione GRAPHIC CLR in particolare consente di usare per i normali programmmi uno spazio di memoria in precedenza riservato alla grafica

**GSHAPE var$**[,x,y,z] - Visualizza sullo schermo la forma grafica memorizzata nella variabile stringa var$. Se non si specifica una data posizione, la forma viene collocata col vertice estremo sinistro in alto nella posizione attuale del cursore pixel. In alternativa, si possono indicare le coordinate che deve assumere il vertice citato (x,y). Il valore z è pure opzionale ed indica il modo di rimpiazzo

GOTO n.o di linea - Forza il computer a proseguire l'esecuzione dall'istruzione di dato numero di linea

GOSUB n.o di linea - Come GOTO, ma qui il comando fa saltare il computer ad eseguire una subroutine del programma che inizia con un certo numero di linea. Quando in questa si incontra l'istruzione RETURN, il computer riprende l'esecuzione dall'istruzione immediatamente successiva al GOSUB

HEADER"nome"[1,Iid, Ddrive, ON Uunita] - Serve per formattare un disco, cosa essenziale per poter usare un disco nuovo. Se viene tralasciato il numero di identificazione (Iid) viene eseguita una formattazione rapida, e viene cancellato soltanto il "catalogo" del disco (equivalente a cancellare tutti i programmi del disco). In questo caso il disco doveva essere stato già formattato in precedenza

HELP - Se in un programma BASIC è intervenuto un errore, questo comando visualizzerà la linea che ha provocato l'errore, con il punto "incriminato" lampeggiante

HEX$(n) - Converte il numero decimale n in esadecimale

IF arg.Booleano THEN istruzioni [:ELSE istruzioni] - Le istruzioni che seguono THEN verranno eseguite se la condizione espressa dall'argomento Booleano è verificata. Se no, verranno eseguite le istruzioni

poste dopo ELSE, oppure quelle immediatamente successive se manca la parte ELSE

INPUT ["stringa";]1 var1, var2,var3,... - Visualizza la stringa (se questa è presente nel comando) ed attende l'impostazione di un dato dalla tastiera, che verrà poi assegnato alla variabile var1 (se seguono più variabili, si possono introdurre di seguito altrettanti dati). Le variabili ed i dati impostati possono essere secondo i casi reali, intere o stringa

INPUT# numero file, var, var,var,... - Simile ad INPUT salvo che i dati vengono ricavati dal file fornito da una periferica

INSTR(stringa1,stringa2 [,iniz] - Ritorna la posizione occupata dal primo carattere di stringa2 entro la stringa1. Se è specificato iniz, la ricerca parte dalla relativa posizione entro stringa1

INT(x) - Ritorna il valore di x arrotondato all'intero per difetto.

JOY(x) - Ritorna un valore che dipende dalla posizione assunta dal joystick x. Se è premuto il pulsante "FIRE" al valore viene aggiunto 128

KEY[n,stringa comandi] - Visualizza i comandi assegnati al tasto funzione n. Con lo stesso comando il tasto può venire ridefinito

LEFT$(stringa,n) - Ritorna la stringa costituita dagli ultimi n caratteri della "stringa"

LEN(stringa) - Ritorna il numero di caratteri (inclusi gli spazi) che sono compresi in "stringa"

LET var=valore - Può essere impiegato per assegnare un valore ad una variabile, ma non è necessario

LIST[inizio-fine] - Visualizza tutto, o parte del listato di un programma sullo schermo

LOAD"nome programma"[,periferica,rilocazione] - Ricarica in memoria il programma "nome programma" dal nastro. Se è specificato il valore "rilocazione" come 1, il programma viene ricaricato in memoria a partire dalla locazione da cui era stato prelevato: la cosa si applica però in genere solo ai programmi in linguaggio macchina

LOCATE x,y/d;a - Posiziona il cursore pixel nel punto di coordinate

(alta risoluzione) x,y. In alternativa il cursore può essere spostato rispetto alla posizione iniziale di d pixel, sotto un angolo di a gradi

LOG(x) - Ritorna il valore del logaritmo naturale di x

LOOP - Definisce il termine d'un ciclo DO...LOOP. Vedi DO

**MID$**(stringa$,x,n) - Fornisce gli n caratteri della stringa, a partire dal suo x.mo carattere

MONITOR - Serve per entrare nel programma Monitor TEDMON

NEW - Cancella programma e variabili correntemente presenti in memoria

NEXT[var] - Definisce la fine di un ciclo FOR...NEXT. Vedi FOR

NOT n - Operatore Booleano, che fornisce il valore di NOT n ("n negato"). Si può pure impiegare con IF...THEN (p. es. IF NOT A=1 THEN PRINT"SI" visualizza SI solo se A non è eguale ad 1)

ON n GOTO/GOSUB linea1, linea2,.... - In funzione del valore assunto dalla variabile n si ha un salto ad uno dei numeri di linea della lista che segue

OPEN numerofile, unita [,indir.secondario,"nome file, tipo file, modo"] - Apre un canale per una unità periferica, assegnandole un numero di file

OR - Impiegato con IF...THEN (p. es. IF A=1 OR B=1 THEN STOP provocherà l'arresto del programma se il valore di A è uguale ad 1, oppure B è uguale ad 1, oppure si verificano entrambe le condizioni

PAINT[cs,x,y,modo] - Riempie un'area chiusa di colore. Se non sono indicate le coordinate del punto iniziale, parte dalla posizione corrente del cursore pixel. Se non è definito il colore sorgente, viene usato il colore corrente dello sfondo

PEEK(indirizzo) - Ritorna il contenuto della locazione di memoria di dato indirizzo

POKE indirizzo, valore - Memorizza nella locazione di dato indirizzo il valore numerico specificato

POS(x) - Ritorna la posizione (coordinata) orizzontale corrente del cursore. x può essere qualsiasi (p. es. 0)

PRINT lista - Visualizza sullo schermo i contenuti della "lista" (p. es. PRINT"SALVE" visualizza SALVE; PRINT 4+5 visualizza il risultato della somma 4+5)

PRINT#numero file, lista - Simile a PRINT con la differenza che i dati della lista vengono trascritti nel file di dato numero di file, e non sullo schermo

PUDEF"caratteri" - Serve a ridefinire i simboli definiti con l'istruzione PRINT USING

RCLR(n) - Ritorna il colore correntemente assegnato al colore sorgente tipo n

RDOT(n) - Ritorna informazioni relative al cursore pixel (n=0 per la coordinata x, n=1 per la coordinata y, n=2 per il colore sorgente)

READ var,var,var... - Legge i dati elencati in una lista di DATA, assegnandoli alle variabili nell'ordine in cui sono indicate. var può essere nei vari casi una variabile di qualsiasi tipo

RENAME"vecchio file" TO "nuovo file"[,Ddrive, Uunita] -Assegna un nuovo nome ad un vecchio file su un disco

RENUMBER[nuovo n.o linea,passo,vecchio n.o linea] - Rinumera tutte le linee di un programma a partire dal "vecchio numero di linea" indicato, assegnandogli il 'nuovo n.o di linea e con incremento fra i successivi numeri di linea eguale al passo specificato. Se non sono designati vecchio e nuovo n.o di linea, nè il passo, il programma viene rinumerato assegnando il n.o 10 alla sua prima linea, con passo eguale a 10 fra le linee

REM commento - Serve per inserire dei commenti in un programma, a fini di documentazione. Ogni cosa dopo REM viene trascurata dal computer

RESTORE[n.o di linea] - Rimanda il computer alla lettura dei dati a partire dalla prima linea di DATA del programma. Se però è specificato un numero di linea, la lettura riprende da quella linea o dal DATA immediatamente successivo

RESUME[n.o di linea/NEXT] - Dopo l'esecuzione d'un'istruzione TRAP, questo comando fa ripetere tentativamente l'esecuzione dell'istruzione che ha provocato l'errore. Con RESUME NEXT si dice invece al computer di riprendere dall'istruzione immediatamente successiva a quel-

la che ha causato l'errore. Se viene specificato un numero di linea, è da questo che il computer riprenderà l'esecuzione del programma

RETURN - Istruzione posta al termine di una subroutine richiamata con GOSUB, che dice al computer di tornare all'istruzione immediatamente successiva al GOSUB

RGR(n) - Ritorna il modo grafico corrente. n è arbitrario, e può essere ad es. 0

RIGHT$(stringa, n) - Ritorna gli ultimi n caratteri della stringa

RLUM(cs) - Ritorna il livello di luminosità del colore sorgente cs

RND(base) - Ritorna un valore a caso fra 0 ed 1. Il modo con cui il numero viene scelto dipende dal valore della "base"

RUN [n.o di linea] - Fa partire l'esecuzione dal programma in memoria.: se è indicato un numero di linea, l'esecuzione principia da questo

SAVE"nome programma"[,numero unita,EOT] - Salva su nastro - a meno che non sia specificato un altro numero di unità periferica - il programma in memoria, assegnandoli il "nome programma". Se alla fine si trova ,1 viene aggiunto un marcatore di 'fine nastrò (EOT)

SCALE 1/0 - Attiva o disattiva la modalità di scala

SCNCLR - "Pulisce" (cancella interamente) lo schermo

SCRATCH"nome file"[,Ddrive, Uunita] - Cancella il dato programma da un disco

SGN(n) - Ritorna il valore -1, 0 od 1 a seconda che n sia negativo, nullo o positivo

SIN(a) - Ritorna il valore del seno dell'angolo a (in radianti)

SOUND voce,valore,durata - Emette una nota di dato "valore" dalla data "voce" per la "durata" indicata

SPC(n)- Simile a TAB, con la differenza che funziona pure con la stampante, cosa che non vale per TAB

SQR(n) - Ritorna il valore della radice quadrata del numero n

**SSHAPE var$**,x1,y1[,x2,y2] - Memorizza una certa area dello schermo dalle coordinate x1,y1 sino a x2,y2 nella variabile stringa var$. Se x2 ed y2 non sono specificati si assumono i valori correnti del cursore pixel

STEP incremento - Usato con FOR...NEXT. Vedi FOR

STOP - Ferma l'esecuzione del programma, con messaggio BREAK

STR$(n) - Converte il numero n in una stringa

SYS indirizzo - Fa eseguire il programma in linguaggio macchina che parte dal dato indirizzo di memoria

TAB(n) - Sposta il cursore sulla colonna n 1 dello schermo

TAN(a) - Ritorna il valore della tangente di un angolo (espresso in radianti)

THEN istruzioni - Parte di un'istruzione IF...THEN...ELSE. Vedi IF

TO - Usato nei cicli FOR (vedi)

TRAP nl - Fa saltare il computer alla linea nl del programma se interviene un errore

TROFF - Disattiva la modalità "tracciamento" durante l'esecuzione

TRON - Attiva il "tracciamento" dell'esecuzione del programma

UNTIL arg. Booleano - Usato nei cicli DO...LOOP. Vedi DO

USING stringa; lista - Usato nelle istruzioni PRINT per specificare il formato di stampa con cui devono essere stampati i valori della lista

var=USR(x) - Serve a lanciare l'esecuzione d'un programma in linguaggio macchina, a partire dall'indirizzo di memoria contenuto nelle locazioni 1281/2. Il valore di x viene passato al L. M. nell'accumulatore in virgola mobile. La variabile var, al rientro al BASIC, conterrà un valore passato dal L. M. al BASIC

VAL(stringa) - Ritorna il valore numerico della stringa

VERIFY"nome programma" [,periferica, flag di rilocazione] -Confronta il programma memorizzato sul nastro - a meno che non sia indica-

to un diverso numero di periferica - con quello in memoria. Se sono identici compare il messaggio OK; altrimenti VERIFY ERROR

VOL v - Fissa a v il livello del volume sonoro

WAIT indirizzo, valore1 [,valore2] - L'esecuzione del programma si arresta sino al momento in cui il valore contenuto nel dato indirizzo di memoria, di cui si esegue l'EXOR con valore2 e quindi l'AND con valore1, fornisce un risultato diverso da 0

WHILE arg.Booleano - Usato nei cicli DO...LOOP. Vedi DO

Abbreviazioni dei comandi BASIC

La maggior parte dei comandi, istruzioni e funzioni che il vostro computer è in grado di capire si possono fornire in forma abbreviata. Generalmente, si tratta delle prime due o tre lettere del comando, seguiti da un ultimo carattere premuto in modo SHIFT (ossia premendo SHIFT mentre si preme il tasto del carattere). Una lista di tali abbreviazioni è data qui di seguito: il carattere che è indicato fra parentesi quadre è quello che va battuto assieme al tasto SHIFT

ABS	A[B]	A│	GET	G[E]	G▔
ASC	A[S]	A♥	GETKEY	GETK[E]	GETK▔
ATN	A[T]	A│	GET#	NONE	GET#
AUTO	A[U]	A⟋	GOSUB	GO[S]	GO♥
BACKUP	B[A]	B♠	GOTO	G[O]	G┌
BOX	B[O]	B┌	GRAPHIC	G[R]	G▁
CHAR	CH[A]	CH♠	GSHAPE	G[S]	G♥
CHR$	C[H]	C│	HEADER	HE[A]	HE♠
CIRCLE	C[I]	C⟍	HELP	HE[L]	HEL
CLOSE	CL[O]	CL┌	HEX$	H[E]	H▔
CLR	C[L]	CL	IF	NONE	IF
CMD	C[M]	C╲	INPUT	NONE	INPUT
COLLECT	COL[L]	COLL	INPUT#	I[N]	I╱
COLOR	CO[L]	COL	INSTR	IN[S]	I╱
CONT	C[O]	C┌	INT	NONE	INT
COPY	CO[P]	CO┐	JOY	J[O]	J┌
COS	NONE	COS	KEY	K[E]	K▔
DATA	D[A]	D♠	LEFT$	LE[F]	LE▁
DEC	NONE	DEC	LEN	NONE	LEN
DEF	D[E]	D▔	LET	L[E]	L▔
DELETE	DE[L]	DEL	LIST	L[I]	L⟍
DIM	D[I]	D⟍	LOAD	L[O]	L┌
DIRECTORY	DI[R]	DI▁	LOCATE	LO[C]	LO▔
DLOAD	D[L]	DL	LOG	NONE	LOG
DO	NONE	DO	LOOP	LO[O]	LO┌
DRAW	D[R]	D▁	MID$	M[I]	M⟍
DSAVE	D[S]	D♥	MONITOR	M[O]	M┌
ELSE	E[L]	EL	NEW	NONE	NEW
END	E[N]	E╱	NEXT	N[E]	N▔

ERR$	E[R]	E_	NOT	N[O]	N┌
EXIT	EX[╲]	EX╲	ON	NONE	ON
EXP	E[X]	E♣	OPEN	O[P]	O┐
FOR	F[O]	F┌	PAINT	P[A]	P♠
FRE	F[R]	F_	PEEK	P[E]	P‾
POKE	P[O]	P┌	SOUND	S[O]	S┌
POS	NONE	POS	SPC	S[P]	S┐
PRINT	?	?	SQR	S[Q]	S●
PRINT#	P[R]	P_	SSHAPE	S[S]	S♥
PUDEF	P[U]	P╭	ST	NONE	ST
RCLR	R[C]	R─	STEP	ST[E]	ST‾
RDOT	R[D]	R‾	STOP	S[T]	S|
READ	R[E]	R‾	STR$	ST[R]	ST_
REM	NONE	REM	SYS	S[Y]	S|
RENAME	RE[N]	RE╱	TAB	T[A]	T♠
RENUMBER	REN[U]	REN╭	TAN	NONE	TAN
RESTORE	RE[S]	RE♥	THEN	T[H]	T|
RESUME	RES[U]	RES╭	TI	NONE	TI
RETURN	RE[T]	RE|	TI$	NONE	TI$
RGR	R[G]	R|	TRAP	T[R]	T_
RIGHT$	R[I]	R╲	TROFF	TRO[F]	TRO_
RLUM	R[L]	R└	TRON	TR[O]	TR┌
RND	R[N]	R╱	UNTIL	U[N]	U╱
RUN	R[U]	R╭	USING	US[I]	US╲
SAVE	S[A]	S♠	USR	U[S]	U♥
SCALE	SC[A]	SC♠	VAL	NONE	VAL
SCNCLR	S[C]	S─	VERIFY	V[E]	V‾
SCRATCH	SC[R]	SC_	VOL	V[O]	V┌
SGN	S[G]	S|	WAIT	W[A]	W♠
SIN	S[I]	S╲	WHILE	W[H]	W|

Codici CHR$ dei caratteri

Ecco una lista che elenca i vari codici CHR$ per tutti i caratteri disponibili sul vostro computer. Come potete vedere, alcuni non hanno effetto, ed altri sono presenti in doppio.

```
 0
 1
 2
 3
 4
 5 WHITE
 6
 7
 8 DISABLES SHIFT-C=
 9 ENABLES SHIFT-C=
10
11
12
13 CARRIAGE RETURN
14 LOWER-CASE
15
16
17 CURSOR DOWN
18 REVERSE ON
19 CLEAR SCREEN
20 DELETE
21
22
23
24
25
26
```

```
31 BLUE
32 SPACE
33 !
34 "
35 #
36 $
37 %
38 &
39 '
40 (
41 )
42 *
43 +
44 ,
45 -
46 .
47 /
48 0
49 1
50 2
51 3
52 4
53 5
54 6
55 7
56 8
57 9
58 :
```

27
28 RED
29 CURSOR RIGHT
30 GREEN
63 ?
64 @
65 A
66 B
67 C
68 D
69 E
70 F
71 G
72 H
73 I
74 J
75 K
76 L
77 M
78 N
79 O
80 P
81 Q
82 R
83 S
84 T
85 U
86 V
87 W
88 X
89 Y
90 Z
91 [
92 £
93]
94 ↑
95 ←
96 —
97 ♠
98 |

59 ;
60 <
61 =
62 >
106 ╮
107 ╯
108 ·L
109 ╲
110 ╱
111 ┌
112 ┐
113 ●
114 _
115 ♥
116 |
117 ╭
118 ×
119 ○
120 ♣
121 |
122 ♦
123 +
124 ▒
125 |
126 π
127 ◥
128
129 ORANGE
130
131
132
133
134
135
136
137
138
139
140
141 SHIFT-RETURN

99 ─
100 ¯
101 ¯
102 _
103 |
104 |
105 ◥

149 BROWN
150 YELLOW GREEN
151 PINK
152 BLUE GREEN
153 LIGHT BLUE
154 DARK BLUE
155 LIGHT GREEN
156 PURPLE
157 CURSOR LEFT
158 YELLOW
159 CYAN
160 SPACE
161 ▌
162 ▄
163 ▔
164 ▁
165 ▏
166 ▒
167 ▕
168 ▒
169 ◤
170 ▕
171 ├
172 ▗
173 └
174 ┐
175 ▂
176 ┌
177 ┴
178 ┬
179 ┤
180 ▎
181 ▍

142 UPPER-CASE
143
144 BLACK
145 CURSOR UP
146 REVERSE OFF
147 CLEAR SCREEN
148 INSERT

192 ─
193 ♠
194 │
195 ─
196 ─
197 ─
198 ─
199 │
200 │
201 ╮
202 ╰
203 ╯
204 L
205 ╲
206 ╱
207 ┌
208 ┐
209 ●
210 ─
211 ♥
212 │
213 ╭
214 ╳
215 ○
216 ♣
217 │
218 ♦
219 ┼
220 ▒
221 │
222 π
223 ◥
224

182 ▐
183 ▔
184 ▀
185 ▄
186 ┘
187 ▖
188 ▝
189 ┘
190 ▘
191 ▚
235 ├
236 ▪
237 └
238 ┐
239 ▄
240 ┌
241 ┴
242 ┬
243 ┤
244 ▏
245 ▌
246 ▐
247 ▔
248 ▀
249 ▄
250 ┘
251 ▖
252 ▝
253 ┘
254 ▘
255 π

225 ▌
226 ■
227 ▔
228 ▁
229 ▏
230 ▒
231 ▕
232 ░
233 ◤
234 ▕

Codici di schermo

Viene fornita qui di seguito la lista dei primi 128 caratteri e dei relativi codici di schermo. Gli altri 128 successivi sono identici ai primi, nell'ordine, salvo che sono a caratteri invertiti. Così se desiderate una lettera 'H' a campo invertito, basta aggiungere 128 al codice di schermo di H (che è 8) per avere quello della H invertita (che è 136).

0	@	20	T	40	(	60	<	80	┐	100	_
1	A	21	U	41	)	61	=	81	●	101	▏
2	B	22	V	42	*	62	>	82	–	102	▒
3	C	23	W	43	+	63	?	83	♥	103	▕
4	D	24	X	44	,	64	–	84	▏	104	▒
5	E	25	Y	45	-	65	♠	85	╭	105	◤
6	F	26	Z	46	.	66	│	86	×	106	▕
7	G	27	[	47	/	67	–	87	O	107	├
8	H	28	£	48	0	68	–	88	♣	108	▗
9	I	29	]	49	1	69	‾	89	▕	109	└
10	J	30	↑	50	2	70	_	90	♦	110	┐
11	K	31	←	51	3	71	▏	91	+	111	
12	L	32		52	4	72	▕	92	▒	112	┌
13	M	33	!	53	5	73	╮	93	│	113	┴
14	N	34	"	54	6	74	╰	94	π	114	┬
15	O	35	#	55	7	75	╯	95	◥	115	┤
16	P	36	$	56	8	76	L	96		116	▏
17	Q	37	%	57	9	77	\	97	▌	117	▌
18	R	38	&	58		78	╱	98	▄	118	▐
19	S	39	'	59	;	79	┌	99	‾	119	‾

120 ▔

121 ▬

122 ┘

123 ▪

124 ▪

125 ┘

126 ▪

127 ▚

APPENDICE E

Glossario

Vi capiterà certamente, leggendo articoli su riviste o altri libri sui computer, di incontrare dei termini che non conoscete. Per darvi una mano a capire questo "gergo informatichese" vi diamo un elenco dei termini più comuni con il loro significato.

Accoppiatore acustico - dispositivo collegabile al computer ed in cui può venire inserito un ricevitore telefonico. In questo modo il computer è in grado di comunicare con un altro tramite la linea telefonica.

Alta risoluzione (HRG) - la risoluzione è il numero massimo di pixel rappresentabili sullo schermo: tanto maggiore questo numero, tanto migliore la risoluzione (definizione).

(Linguaggio) Assembly - un linguaggio di programmamazione che utilizza istruzioni simboliche per svolgere certe procedure modificando i contenuti della memoria del computer

BASIC - abbreviazione di Beginners All-purpose Symbolic Instruction Code = Codice di istruzioni simboliche per usi generali per principianti. È il linguaggio maggiormente diffuso e compreso dai microcomputer odierni, incluso il vostro.

Bit - l'unità elementare di informazione trattabile da un computer: può assumere solo i valori 0 ed 1.

Bug - un termine pittoresco ("pulce" o "baco") per definire un errore presente nel programma che non lo fa funzionare, o almeno non regolarmente

Byte - numero binario formato da 8 bit. Dato che sono possibili 2↑8 = 256 combinazioni di 0 e di 1, un byte può rappresentare un numero fra 0 e 255

(Set di) Caratteri - l'insieme di numeri, lettere e simboli che il computer può visualizzare sul normale schermo per i testi

Cartridge - o “cartuccia”: piccolo contenitore che contiene memorizzato un programma, o della RAM extra, e che può essere inserito in un computer.

Codice sorgente - programma scritto in un linguaggio di alto livello, come il BASIC o Pascal, che va compilato a linguaggio macchina prima che possa venire eseguito dalla CPU.

Comando - un'istruzione normalmente impostata direttamente dalla tastiera, senza uso di numero di linea (come RUN, LIST, NEW, ecc.).

Compilatore - un particolare programma, di solito in L. M., che converte un programma in linguaggio 'evoluto', come il BASIC, direttamente in linguaggio macchina.

CP/M = Control Program for Microcomputers - Sistema operativo “standard” per i dischi, disponibile per molti computer basati sul microprocessore Z80. Si tratta di un linguaggio standard, diffuso particolarmente nel settore dei computer “gestionali”. Data la sua standardizzazione, programmi scritti in CP/M sono facilmente trasferibili da un computer ad un altro.

CPU = Central Processing Unit - come dice il nome, è l'unità di elaborazione centrale che costituisce il “cervello” del computer, ed è un tipo speciale di ROM.

Cursore - simbolo che indica dove apparirà sullo schermo il carattere successivo

Dati - termine spesso usato per “informazioni”

Debugging - “spulciamento”: ricerca ed eliminazione degli errori del programma.

Disco - disco di materiale plastico rivestito di uno strato magnetizzabile ed inserito in un involucro protettivo. Programmi e dati possono venire registrati su dischi e poi riletti ad alta velocità. Un disco può memorizzare grandi quantità di informazioni.

DOS = Disk Operating System - Programma contenuto nel computer od entro la unità disco (talvolta deve venire appositamente caricato in memoria) che controlla le operazioni inerenti al disco.

EEPROM = Electrically Erasable Programmable Read Only Me-

mory, ossia ROM programmabile cancellabile per via elettrica. Uno speciale tipo di ROM che può venire riprogrammata, previa cancellazione del contenuto originale mediante impulsi elettrici.

EPROM = Erasable Programmable Read Only Memory, ossia memoria di sola lettura (ROM) programmabile e cancellabile. Simile alla precedente, ma qui la cancellazione avviene per azione delle radiazioni ultraviolette.

Floppy disk - disco flessibile o dischetto

Funzione - istruzione che assume uno o più valori e li elabora in certi modi assegnati, fornendo un risultato.

Hard copy - copia permanente a stampa delle informazioni, prodotta da una stampante.

Hard disk - tipo particolare di disco, in cui il disco rimane fisso entro l'unità disco. È in grado in genere di memorizzare una quantità di informazioni molto maggiore dei soliti Floppy Disk, ma ad un prezzo alquanto superiore, dovuto in prevalenza alla gran cura con cui occorre che la testina di lettura resti allineata sopra il disco.

Hardware - termine intraducibile, con cui si intende l'insieme di tutte quelle parti che compongono "fisicamente" il computer, che potete toccare.

Esadecimale (Hex) - sistema di numerazione in base 16 anzichè 10 come il solito decimale. Per indicare le cifre dopo 0 - 9 si usano le lettere da A ad F (A=10, B=11, C=12,ecc.).

Indirizzo - numero che specifica una locazione di memoria

I/O - sta per Input/Output, ossia Ingresso/Uscita. Riferito ai dati, o ad una periferica che riceve o trasmette informazioni, o ad una 'porta' tramite la quale si possono ricevere o trasferire dei dati.

Istruzione - un termine o gruppo di termini che "istruisce" il computer a fare qualcosa

Kilobyte (K) - 1024 byte di memoria

Linguaggio Macchina (L.M.) - il linguaggio che la CPU è in grado di capire. È costituito interamente da codici numerici!

Mappa di memoria - tabella che indica la disposizione entro la memoria

delle varie aree impegnate dal computer per i vari scopi.

Modem - sta per MOdulatore e DEModulatore: dispositivo che consente al computer di ricevere e trasmettere dati attraverso le linee telefoniche. Per collegare un modem si richiede in genere un'autorizzazione dai Servizi Telefonici competenti.

Modulatore video - dispositivo elettronico normalmente già presente nel computer, che permette di convertire i segnali in una forma visualizzabile su di un ricevitore televisivo.

Monitor - un apparecchio televisivo particolare che fonisce immagini di qualità molto migliore dei normali TV; oppure programma che permette di esaminare ed alterare i contenuti delle varie locazioni di memoria.

Parallelo - uno dei modi con cui il computer riceve o trasmette informazioni. Un'interfaccia parallela consente di trattare un intero gruppo (di solito 8) di bit per volta.

Pascal - un linguaggio di "alto livello" molto potente, usato su certi computer per scopi gestionali.

Periferica (unità) - un'unità o dispositivo che si può collegare al computer.

Pixel - sta per "picture element", ovvero elemento minimo dell'immagine: il punto di dimensioni minime che può venire "illuminato" sullo schermo.

Porta - collegamento attraverso il quale si può introdurre o trasmettere informazioni.

Printout - lo stesso che "Hard copy".

Programma - una serie di istruzioni che svolgono un certo compito, che il computer esegue normalmente in ordine successivo (di numero di linea)

PROM = Programmable Read Only Memory: ossia uno speciale tipo di ROM che può venire opportunamente riprogrammato tramite dispositivi particolari.

QWERTY - così è popolarmente designata la forma di tastiera americana (dai primi cinque tasti letterali: quella europea è detta QZERTY).

RAM = Random Access Memory ossia memoria ad accesso casuale. I

suoi contenuti non sono permanenti, e possono quindi venire modificati. I contenuti di questo tipo di memoria di norma si conservano solo finchè l'alimentazione è inserita, e si perdono se si spegne il computer.

Registro - una parte della CPU che agisce da puntatore per uno specifico blocco della memoria.

ROM = Read Only Memory: memoria di sola lettura, di tipo permanente, per cui non è possibile alterarne il contenuto. Non richiede alimentazione elettrica perchè i suoi contenuti siano conservati.

Routine - una parte di programma destinata a svolgere un compito particolare.

RS232 - una forma standardizzata di interfaccia destinata alla trasmissione seriale delle informazioni

Seriale - l'altro modo con cui si possono ricevere e trasmettere informazioni, un bit alla volta.

Software - per contrapposto all'hardware, la parte "intangibile" che fa funzionare il computer: in genere un programma, memorizzato in qualche modo sull'hardware.

Statement - termine con cui si riferisce ad una sottoistruzione in un'istruzione multi:linea di un programma

Stringa - gruppo di caratteri

Subroutine - vedi routine

Toolkit - programma che aggiunge altri comandi utili a quelli già previsti dal computer.

Utility - routine di utilità: di solito una di quelle che possono comporre un toolkit, od anche separata, per svolgere compiti particolari, specie in fase di impostazione e modifiche ad un programma.

Variabile - un valore che si può modificare, solitamente indicato con un carattere o gruppo di caratteri ("nome").

VDU = Visual Display Unit - unità di visualizzazione (TV o monitor).

Z80 - uno dei più diffusi microprocessori (CPU), usato ad esempio da ZX

Spectrum, computer Tandy (TRS80), Sharp e VideoGenie.

6502 - altro microprocesore molto usato sui microcomputer, e su una derivazione del quale si basa la CPU del vostro computer.

Alcuni esempi di programmi

Per finire, vi presentiamo alcuni programmi non troppo lunghi,da sperimentare

Disegno in 3-D

```
 10 GRAPHIC3,1:COLOR3,2,7
 20 A=80:B=A*A:C=100:D=100
 30 FORX=0TOA
 40 S=X*X
 50 P=SQR(B-S)
 60 I=-P
 70 R=SQR(S+I*I)/A
 80 Q=(R-1)*SIN(24*R)
 90 Y=I/3+Q*D
100 IFI=-PTHENM=Y:GOTO130
110 IFY>MTHENM=Y:GOTO140
120 IFY>=NTHENGOTO170
130 N=Y
140 Y=C+Y
150 DRAW3,A+X,Y
160 DRAW3,A-X,Y
170 I=I+4
180 IFI<PTHEN70
190 NEXTX
200 FORC=2TO15:FORI=0TO7:COLOR3,C,I
210 FORM=0TO500:NEXTM,I,C
220 GOTO200
```

Orologio con allarme

Se date il RUN a questo programma, vi verrà chiesto di indicare l'ora a cui volete scatti l'allarme. Questo programma attiva un orologio in formato 24 ore, per cui l'ora dell'allarme va specificata come HHMMSS,

dove HH sono le ore (2 cifre), MM i minuti ed SS i secondi. Per mettere all'ora l'orologio tenete abbassato il tasto H per muovere la lancetta delle ore, poi la M per i minuti ed infine la S per la lancetta dei secondi.

```
 10 SCNCLR:PRINT"INDICATE L'ORA A CUI VOLET
    E SCATTI       L'ALLARME (HHMMSS)";
 20 INPUTAL$:IFLEN(AL$)<>6THENRUN
 30 VOL5:V=5
 40 Z=350
 50 GRAPHIC1,1
 60 COLOR0,3,3:COLOR4,3,3:COLOR1,8,6
 70 SCNCLR
 80 CIRCLE1,160,100,75
 90 FORN=0TO360STEP6
100 DRAW0,160,100 TO 72;N
110 DRAW1 TO 3;N
120 NEXT
130 FORN=0TO360STEP30
140 DRAW0,160,100 TO 65;N
150 DRAW1 TO 10;N
160 NEXT
170 X=344:GOSUB430
180 Z=344:GOSUB370
190 TI$="000000"
200 IFZ>=704THENZ=344:GOSUB370:GOSUB430
210 FORA=0TO354STEP6
220 SOUND2,1000,1
230 COLOR1,2,7
240 DRAW1,160,100 TO 56;A
250 GOSUB400:GOSUB450
260 GETA$
270 IFA$="C"THENGOSUB470
280 IFA$="A"THENAL=0
290 IFTI$=AL$THENAL=1
300 IFAL=1THENFORM=1TO9:SOUND1,700,2:SOUND1
    ,800,2:NEXT
```

```
310 IFAL=1THEN330
320 FORM=1TO310:NEXTM
330 DRAW0,160,100 TO 56;A
340 NEXTA
350 GOSUB370
360 GOTO200
370 DRAW0,160,100 TO 40;Z TO 18;Z+34 TO 18;
    Z-194 TO 160,100
380 Z=Z+6
390 E=E+1
400 DRAW1,160,100 TO 40;Z TO 18;Z+34 TO 18;
    Z-194 TO 160,100
410 IFE=12THENE=0:GOSUB430
420 RETURN
430 DRAW0,160,100 TO 30;X TO 14;X+33 TO 14;
    X-192 TO 160,100
440 X=X+6
450 DRAW1,160,100 TO 30;X TO 14;X+33 TO 14;
    X-192 TO 160,100
460 RETURN
470 GETKEY A$
480 IFA$="H"THENGOSUB430:Z$=STR$(VAL(TI$)+1
    00000):Z$=RIGHT$(Z$,LEN(Z$)-1)
490 IFA$="M"THENGOSUB370:Z$=STR$(VAL(TI$)+1
    00):Z$=RIGHT$(Z$,LEN(Z$)-1)
500 IFA$="S"THENDRAW0,160,100 TO 56;A:Z$=ST
    R$(VAL(TI$)+1)
510 DRAW0,160,100 TO 57;A

520 IFA$="S"THENZ$=RIGHT$(Z$,LEN(Z$)-1):A=A
    +6
530 DRAW1,160,100 TO 57;A:GOSUB450:GOSUB400
540 IFZ$=""THEN570
550 IFLEN(Z$)<6THENZ$=RIGHT$("000000",6-LEN
    (Z$))+Z$:TI$=Z$:Z$=""
560 IFA$="T"ANDV=0THENVOL5:V=5:ELSEIFA$="T"
    THENV=0:VOL0
570 IFA$="C"THENDRAW0,160,100 TO 57;A:RETUR
    N
580 GOTO470
```

Figure

```
10 GRAPHIC2,1
20 INPUT"QUANTE FACCE";A
30 IFA<2ORA>100THENPRINT"NON ESSERE RIDICO
   LO!":GOTO20
35 SCNCLR
40 CIRCLE1,160,80,40,33,,,,360/A
50 GOTO20
```

ISTRUZIONI PER IL CORRETTO CARICAMENTO DEI PROGRAMMI SU CASSETTA PER COMMODORE C 16 e PLUS 4

Riavvolgete il nastrodella cassetta fino all'inizio.
Digitare LOAD o LOAD"nome programma" e premere il tasto RETURN.
Avviare il registratore con il tasto play ed attendere l'avvenuto caricamento del programma.
Se il programma non è provvisto di AUTOSTART (partenza automatica), digitare RUN seguito dal tasto RETURN.
Pur eseguendo le istruzioni sopraindicate, è possibile incontrare talvolta qualche difficoltà nel caricamento del programma. La prima cosa da fare è assicurarsi che la testina del vostro registratore risulti ben pulita, allineata e smagnetizzata.
Non modificate per nessun motivo l'allineamento della testina del registratore poiché risulta pressoché impossibile, se non si dispone di apparecchiature professionali, procedere alla taratura dello stesso; in tal caso rivolgetevi presso un laboratorio specializzato.

PROGRAMMI CONTENUTI NELLA CASSETTA

Nella cassetta allegata al libro sono presenti i seguenti programmi:

Trap test (7)
Dr. Foster (8)
Telefono (8)
Dado (8)
Cerchi (9)
Pallina (9)
Artista (PLUS 4)
Orologio (F)
Figure (F)

GARANZIA CASSETTA SOFTWARE

Le cassette software che presentano eventuali difetti, non manomesse, vanno spedite, per la sostituzione a:
EDIZIONI JCE
Via dei Lavoratori, 124
20092 Cinisello Balsamo (MI)

MILLECANALI
SELEZIONE
di elettronica e microc
Sperimentare
Computer
EG
COMPUTER
Elettronica Giovani
ANTEPRIMA LAS VEGAS
MSX

PUBBLICAZIONI JCE 1985

LIBRI DI INFORMATICA

DESCRIZIONE	CODICE	PREZZO UNITARIO
IL LIBRO DEL MICRODRIVE SPECTRUM	9001	L. 16.000
FORTH PER SPECTRUM	9005	L. 15.000
ALLA SCOPERTA DEL QL IL COMPUTER SINCLAIR	9050	L. 20.000
COME PROGRAMMARE IL TUO IBM PC	9200	L. 20.000
LA PRIMA VOLTA CON APPLE	9300	L. 16.000
ALLA SCOPERTA DELL'APPLE *IIc*	9301	L. 16.000
APPLE MACINTOSH: IL COMPUTER MAGICO	9350	L. 20.000
ATARI SERIE XL	9411	L. 16.000
IL 68000: PRINCIPI E PROGRAMMAZIONE	9850	L. 20.000

LIBRI DI INFORMATICA CON CASSETTA

DESCRIZIONE	CODICE	PREZZO UNITARIO
SINCLAIR ZX SPECTRUM: ASSEMBLER E LINGUAGGIO MACCHINA	9000	L. 25.000
PROGRAMMARE IMMEDIATAMENTE LO SPECTRUM	9002	L. 25.000
CREATE GIOCHI ARCADE COL VOSTRO SPECTRUM	9003	L. 25.000
APPROFONDIRE LA CONOSCENZA DELLO SPECTRUM	9004	L. 30.000
PROGRAMMIAMO INSIEME LO SPECTRUM	9006	L. 30.000
BASIC & FORTRAN PER SPECTRUM	9007	L. 25.000
POTENZIATE IL VOSTRO SPECTRUM	9008	L. 30.000
49 GIOCHI ESPLOSIVI PER LO SPECTRUM	9009	L. 30.000
GRAFICA AVANZATA CON LO SPECTRUM	9010	L. 35.000
GRAFICA E SUONO PER IL LAVORO E IL GIOCO CON LO SPECTRUM	9011	L. 25.000
METTETE AL LAVORO IL VOSTRO VIC 20	9100	L. 25.000
IL MIO COMMODORE 64	9150	L. 25.000
COME PROGRAMMARE IL TUO COMMODORE 64	9151	L. 25.000
COMMODORE 64: I SEGRETI DEL LINGUAGGIO MACCHINA	9152	L. 30.000
SPRITES & SUONO DEL COMMODORE 64	9153	L. 30.000
SONY MSX BASIC	9400	L. 30.000
IMPARIAMO IL PASCAL SUL NOSTRO COMPUTER	9800	L. 25.000

SOFTWARE

DESCRIZIONE		CODICE	PREZZO UNITARIO
GRAFICA PER TUTTI	SPECTRUM 48k e PLUS	J/0100-01	L. 25.000
MANUALE DI GEOMETRIA PIANA	SPECTRUM 48k e PLUS	J/0100-02	L. 25.000
MANUALE DI GEOMETRIA SOLIDA	SPECTRUM 48k e PLUS	J/0100-03	L. 25.000
TRIGONOMETRIA	SPECTRUM 48k e PLUS	J/0100-04	L. 25.000
MOSAICO	SPECTRUM 48k e PLUS	J/0101-01	L. 20.000
BATTAGLIA NAVALE	SPECTRUM 48k e PLUS	J/0101-02	L. 20.000
PUZZLE MUSICALE	SPECTRUM 48k e PLUS	J/0101-03	L. 20.000
SUPER EG	SPECTRUM 48k e PLUS	J/0101-04	L. 20.000
SPECTRUM WRITER (MICRODRIVE COMPATIBILE)	SPECTRUM 48k e PLUS	J/0102-01	L. 40.000
MASTER FILE (MICRODRIVE COMPATIBILE) VERSIONE ITAL.	SPECTRUM 48k e PLUS	J/0102-02	L. 40.000
BUSINESS GRAPHICS	SPECTRUM 48k e PLUS	J/0102-03	L. 25.000
INGEGNERIA: PROGRAMMA AD ELEMENTI FINITI	SPECTRUM 48k e PLUS	J/0104-01	L. 30.000
TOPOGRAFIA	SPECTRUM 48k e PLUS	J/0104-02	L. 30.000
CALCOLO TRAVI IPE	SPECTRUM 48k e PLUS	J/0104-03	L. 25.000
ENERGIA SOLARE	SPECTRUM 48k e PLUS	J/0104-04	L. 30.000
ALGEBRA MATRICIALE	SPECTRUM 48k e PLUS	J/0104-05	L. 30.000
STUDIO DI FUNZIONI	SPECTRUM 48k e PLUS	J/0104-06	L. 30.000

SOFTWARE

DESCRIZIONE		CODICE	PREZZO UNITARIO
EQUAZIONI PARAMETRICHE E PROBLEMI DI 2°GRADO	SPECTRUM 48k e PLUS	J/0104-07	L. 25.000
TOTIP	SPECTRUM 48k e PLUS	J/0105-01	L. 20.000
ASTROLOGIA	SPECTRUM 48k e PLUS	J/0105-02	L. 25.000
CAMPIONATO DI CALCIO	SPECTRUM 48k e PLUS	J/0105-03	L. 25.000
RACCOLTA DI QUIZ PER LA PATENTE (MICRODRIVE TRASFERIBILE)	SPECTRUM 48k e PLUS	J/0105-04	L. 25.000
GARDEN WARS	COMMODORE C64	J/0111-01	L. 20.000
ECONOMIA FAMILIARE	COMMODORE C64/DISCO	J/0112-02	L. 40.000
CHESS WARS	COMMODORE VIC 20 NON ESPANSO	J/0121-01	L. 20.000

LIBRI DI ELETTRONICA

DIGIT 1	2000	L. 7.000
CORSO DI PROGETTAZIONE DEI CIRCUITI A SEMICONDUTTORE	2002	L. 8.400
APPUNTI DI ELETTRONICA - VOL. 1	2300	L. 8.000
APPUNTI DI ELETTRONICA - VOL. 2	2301	L. 8.000
APPUNTI DI ELETTRONICA - VOL. 3	2302	L. 8.000
APPUNTI DI ELETTRONICA - VOL. 4	2303	L. 8.000
APPUNTI DI ELETTRONICA - VOL. 5	2304	L. 8.000
APPUNTI DI ELETTRONICA - VOL. 6	2305	L. 8.000
COSTRUIAMO UN VERO MICROELABORATORE ELETTRONICO	3000	L. 4.000
JUNIOR COMPUTER - VOL. 1	3001	L. 11.000
JUNIOR COMPUTER - VOL. 2	3002	L. 14.500
GUIDA ALL'ACQUISTO DEI SEMICONDUTTORI	4000	L. 6.000
TABELLA EQUIVALENZE SEMICOND. E TUBI ELETTRONICI PROFESSIONALI	6006	L. 5.000
TRANSISTOR CROSS-REFERENCE GUIDE	6007	L. 8.000
SELEZIONE DEI PROGETTI ELETTRONICI	6008	L. 9.000
300 CIRCUITI	6009	L. 12.500
THE WORLD TTL, IC DATA CROSS REFERENCE GUIDE	6010	L. 20.000
DIGIT 2	6011	L. 6.000
273 CIRCUITI	6014	L. 12.500
NUOVISSIMO MANUALE DI SOSTITUZIONE FRA TRANSISTORI	6015	L. 10.000
SISTEMI HI-FI MODULARI DA 3 A 1000W	6016	L. 6.000
100 RIPARAZIONI TV ILLUSTRATE E COMMENTATE	7000	L. 10.000
LE RADIO COMUNICAZIONI	7001	L. 7.500
PRATICA TV	7002	L. 10.500
99 RIPARAZIONI TV ILLUSTRATE E COMMENTATE	7003	L. 16.000
ALLA RICERCA DEI TESORI	8001	L. 6.000
LE LUCI PSICHEDELICHE	8002	L. 4.500
ACCESSORI ELETTRONICI PER AUTOVEICOLI	8003	L. 6.000
IL MODERNO LABORATORIO ELETTRONICO	8004	L. 6.000
LA PRATICA DELLE MISURE ELETTRONICHE	8006	L. 11.500

Giugno 1985

Questo libro, che si rivolge a chi è alle prime armi, è stato pensato come manuale per l'uso e guida alla programmazione nel migliore e potenziato BASIC versione 3.5 presente sul Commodore C16 ed anche sul Plus 4. Le caratteristiche di questo BASIC tendono ad evidenziare le possibilità grafiche e di struttura dei due calcolatori Commodore sui quali è stato implementato con successo. Le varie istruzioni del linguaggio BASIC vengono presentate con semplici programmi contenuti nella cassetta allegata, che mettono in grado il lettore di programmare immediatamente il computer. A mano a mano che il lettore acquisterà maggiori conoscenze di programmazione, vengono introdotte istruzioni più complesse. Alla fine della lettura si saranno acquisite anche le tecniche di programmazione più sofisticate, come la gestione dei files e la grafica ad alta risoluzione.

ISBN 88-7708-004-3

Cod. 9115

L. 23.000 LIBRO + CASSETTA